

Management of the Internet and Complex Services

European Sixth Framework Network of Excellence FP6-2004-IST-026854-NoE

Deliverable D1.3 Report on Vision and Integration Program, Including a Handbook of IT Management

The EMANICS Consortium

Caisse des Dépôts et Consignations, CDC, France
Institut National de Recherche en Informatique et Automatique, INRIA, France
University of Twente, UT, The Netherlands
Imperial College, IC, UK
International University Bremen, IUB, Germany
KTH Royal Institute of Technology, KTH, Sweden
Oslo University College, HIO, Norway
Universitat Politècnica de Catalunya, UPC, Spain
University of Federal Armed Forces Munich, CETIM, Germany
Poznan Supercomputing and Networking Center, PSNC, Poland
University of Zürich, UniZH, Switzerland
Ludwig-Maximilian University Munich, LMU, Germany
University of Surrey, UniS, UK
University of Pitesti, UniP, Romania

© Copyright 2007 the Members of the EMANICS Consortium

For more information on this document or the EMANICS Project, please contact:

Dr. Olivier Festor
Technopole de Nancy-Brabois — Campus scientifique
615, rue de Jardin Botanique — B.P. 101
F—54600 Villers Les Nancy Cedex
France

Phone: +33 383 59 30 66
Fax: +33 383 41 30 79
E-mail: <olivier.festor@loria.fr>

Document Control

Title: Report on Vision and Integration Program, Including a Handbook of IT Management

Type: Public

Editor(s): Gabi Dreo Rodosek, Iris Hochstatter

E-mail: gabi.dreo@unibw.de, iris.hochstatter@unibw.de

Author(s): WP1 Partners

Doc ID: D1.3.doc

AMENDMENT HISTORY

Version	Date	Author	Description/Comments
V0.1	June 8, 2007	Iris Hochstatter	Initial version
V0.2	June 12, 2007	Iris Hochstatter	Handbook Preface and contributors
V0.3	June 25, 2007	Gabi Dreo Rodosek, Iris Hochstatter	Summary, handbook chapter abstracts
V0.4	June 27, 2007	Gabi Dreo Rodosek, Iris Hochstatter	Minor changes
V0.5	July 2, 2007	Iris Hochstatter	WP1 partner input
V0.6	July 3, 2007	Gabi Dreo Rodosek, Iris Hochstatter	WP1 partner input
V0.7	July 4, 2007	Gabi Dreo Rodosek, Iris Hochstatter	Introduction, summary
V0.8	July 6, 2007	Gabi Dreo Rodosek, Iris Hochstatter	Overview of Achievements
V0.9	July 9, 2007	Gabi Dreo Rodosek, Iris Hochstatter	Finalization

Legal Notices

The information in this document is subject to change without notice.

The Members of the EMANICS Consortium make no warranty of any kind with regard to this document, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. The Members of the EMANICS Consortium shall not be held liable for errors contained herein or direct, indirect, special, incidental or consequential damages in connection with the furnishing, performance, or use of this material.

Table of Contents

1	Executive Summary	5
2	Introduction	6
3	Overview of Achievements on Various Topics	8
3.1	PhD Students Database	8
3.2	Common Course Program	10
3.3	EMANICS Grants Reflecting Integration and Mobility Support	12
3.4	Towards an Ontology of IT Management	15
3.4.1	Why ontology of IT management?	15
3.4.2	First Steps	16
4	Handbook on Network and Service Administration, published by Elsevier	19
4.1	Table of Contents	19
4.2	Preface	29
5	Summary and Conclusions	31
6	References	32
7	Acknowledgement	32
8	EMANICS contributions to the “Handbook of Network and System Administration”	32

(This page is left blank intentionally.)

1 Executive Summary

Deliverable D1.3 presents a remarkable achievement of the NoE partners with respect to integration, vision and visibility of EMANICS by introducing a Handbook of Network and System Administration with joint contributions from EMANICS partners, published by Elsevier. Although the idea of writing a handbook was documented also in the project plan, it was not intended at that time to publish it by a publisher. However, during the joint discussions on writing a handbook it was decided to do so. The objectives have been as follows:

- to foster integration and collaboration among EMANICS partners by writing together a handbook, to
- increase the visibility of EMANICS by publishing the results and achievements (i.e. by Elsevier) and explicitly acknowledge EMANICS on various places in the handbook, and to
- produce also teaching material since the contributions do not identify only upcoming challenges and a vision for the future in the area of integrated management but they reveal also the current state-of-the-art on some particular topics.

The handbook has been edited by Mark Burgess, EMANICS partner, with the University College Oslo, and Jan Bergstra from University of Amsterdam, and it details on cutting edge topics in IT management like for instance virtualization, economic management, management of ad-hoc networks, and IT service management. Thus, deliverable D1.3 thus presents the handbook and points especially to chapters that have been contributed by EMANICS. Those chapters are also appended.

Although work has continued on other topics of work package 1, as described in the introduction, and will continue in phase 2 on even broader scope, we do not consider this work final. Thus, we slightly changed the title of deliverable D1.3 by deleting the word "Final".

D1.3 reports also on achievements on other topics of WP1 such as the PhD student database and mobility support.

2 Introduction

With this deliverable, the EMANICS NoE is presenting results at the end of the first phase regarding the vision and integration program. We therefore structure this introduction to match the tasks in the initial "Description of Work (DoW)". Many of those tasks will be pursued in the second phase and we will build a bridge between results achieved so far and plans for the second phase.

Task 1.1 Research Observatory

As the partners of the NoE have been jointly working on research topics as well as integration and dissemination activities very closely for the past months, the idea to join forces on writing a handbook that should be published and therefore made available to a large audience was developed. It should combine the latest results and achievements of EMANICS partners in the field of network and service management. The handbook that evolved will be published by Elsevier in 2007 and will increase the visibility of EMANICS greatly, as EMANICS is prominently cited on various places in the book. Chapter 4 provides detailed insight into the handbook by giving all abstracts of chapters contributed by EMANICS partners and citing the introduction to the handbook. Chapters that have been contributed by EMANICS partners are also appended, see chapter 8.

Regarding the identification of new challenges, orientations and visions of network and service management, the handbook delivered a current status of results and vision among EMANICS partners. To identify new challenges of IT management, however, more joint work is needed, and will be also a major topic in the second phase of the NoE. The next event planned will take place already at the end of July 2007 at Dagstuhl, experts from many countries around the world will meet at the seminar "Automatic Management of Networks and Services"

(<http://www.dagstuhl.de/de/programm/kalender/semhp/?semnr=2007302>).

Task 1.2 PhD Integration Program

Chapter 3 provides an overview of results achieved so far on different topics of work package 1. Especially, the PhD integration program has been fostered by

- starting to build a database of PhD students at EMANICS partner institutions (see section 3.1) and to
- encourage the presentation of and discussions about PhD research questions, approaches and results in front of all EMANICS partners, animate discussion, as it succeeded with AIMS 2007.

With this approach, also joint PhD committees are encouraged even stronger. All activities in phase I regarding the PhD integration program have been quite successful and will be continued throughout the second phase.

Task 1.3 Common Course Program & Mobility Support

Work towards a common course program has been accompanied by various discussions how to achieve this ambitious objective due to the fact that it e.g.

- should be applicable for PhD and MSc level, it

- should allow to exchange students among EMANICS partners by allowing to recognize credit points of a module that a MSc student has heard at a partner institution.

Having this in mind, a two-step approach has been chosen:

1. Building an EMANICS teaching material repository by collecting teaching material in various forms (slides, labs, podcasts, etc.) and various languages by adding some meta data to describe the content and form. The repository is accessible to EMANICS partners over the EMANICS web server.
2. Discussions within work package 1 lead to the question whether the proposed module structure of the common course program is the most appropriate form to achieve our objective to foster exchange of students (PhD and MSc), and to achieve a common level of knowledge among partners. An alternative is to build a graduate program in the area of integrated network and service management, including not only IT management related topics but also basics in networking, distributed algorithms, operating system concepts, cryptographic algorithms, logic and formal methods etc.

During the discussions and work package 1 meetings it was recognized that a fundament of further joint activities (research and teaching) is a common understanding. Thus, work on an ontology on IT management has started at the work package 1 meeting in Munich in March. Initial results are presented in chapter 3.4 and work will be strengthened in phase II as the work toward a common ontology on IT Management will be supported as a separate task.

During the last reporting period, mobility and collaboration have increased significantly as presented in our mobility report in section 3.3.

Deliverable 1.3 is structured as follows:

- Chapter 3 present the progress made regarding all tasks specified in the DoW for phase I.
- Chapter 4 focuses on a prominent achievement of the NoE regarding integration, challenges and visions on network and service management, the “Handbook on Network and System Administration”.
- The deliverable concludes with a summary of the work presented at the end of the first phase of the NoE EMANICS and gives an outlook to future work in chapter 5.

3 Overview of Achievements on Various Topics

The following section gives an overview of results and achievements obtained on various topics in work package 1 within the last reporting period.

3.1 *PhD Students Database*

A precondition to establish a long lasting PhD integration program within EMANICS is to build a database of PhD students. So far, an analysis of the relevant information necessary to collect and the technical means has been evaluated and established. We present here the initial collected information. The database of PhD students complements the research map of EMANICS partners as presented in D1.1 since it documents expertise not only associated with institutions but also supervisors and PhD students. Thus, finding adequate supervisors, also with respect to building joint PhD committees and exchanging students, becomes much easier.

It should be stressed that the table below is just an excerpt of the database which is accessible over the EMANICS server. More information like start and approximate end date as well as descriptions of PhD research questions, approaches etc. are accessible over the EMANICS web server. The titles provided should also be considered as working titles.

Table 1: EMANICS PhD Students

Institution	PhD Student	Title	Supervisor
KTH	Fetahi Z. Wuhib	Self-organizing, decentralized management	Rolf Stadler
KTH	Alberto Gonzalez	Adaptive Real-time Monitoring	Rolf Stadler
UniZH	Thomas Bocek	Incentive Management in P2P Systems	Burkhard Stiller
UniZH	Martin Wald-burger	Contract Management in Distributed Systems	Burkhard Stiller
UniZH	Gregor Schaf-frath	Intrusion Detection Systems	Burkhard Stiller
UniZH	Peter Racz	Accounting Management	Burkhard Stiller
UniZH	Cristian Morariu	Accounting Management in P2P Systems	Burkhard Stiller
UniZH	Pascal Kurtansky	Pre-paid Charging in All-IP Networks	Burkhard Stiller
UniZH	Fabio Hecht	tbd	Burkhard Stiller
UT	Tiago Fioreze	Self-management of hybrid optical and packet switching networks	Aiko Pras
UT	Anna Sperotto	Scalable and self-learning Intrusion detection	Aiko Pras
IUB	Ha Manh Tran	Distributed Case-based Reasoning for Fault Management	Juergen Schoenwaelder

Institution	PhD Student	Title	Supervisor
IUB	Iyad Tumar	Management and Scheduling in Disruption Tolerant Networks	Juergen Schoenwaelder
CETIM	Matthias Göhner	A service-oriented accounting system for dynamic Virtual Organizations	Gabi Dreö Rodosek
CETIM	Frank Eyermann	MESA: End-to-End Delay Measurement	Gabi Dreö Rodosek
CETIM	Iris Hochstatter	Accounting in Context-based Systems	Gabi Dreö Rodosek
CETIM	Sameh Bel Haj Saad	Customer Service Management for Parallel and Distributed Simulation running on the GRID	Gabi Dreö Rodosek
UPC	Javier Rubio	A Methodological Approach to Policy Refinement in Policy-based Management Systems	Joan Serrat
UPC	Martín Serrano	Context Information Modeling Based on Ontologies	Joan Serrat
UPC	Edgar Magaña	Heuristic Algorithm for Scheduling Computational Resources in Policy-Based Grid Network Management	Joan Serrat
UniS	Stylianios Georgoulas	Admission Control and Bandwidth Management in IP Differentiated Services Networks	George Pavlou
UniS	Apostolis Malatras	Context-Awareness in Mobile Ad Hoc Networks	George Pavlou
UniS	Aimilios Chourmouziadis	Network Management Using Web Services	George Pavlou
UniS	Marinos Charalambides	Policy Analysis for Quality of Service Management	George Pavlou
UniS	Antonis Hadjiantonis	Policy-based management of ubiquitous systems	George Pavlou
UniS	Mina Amin	Survivability provisioning through traffic engineering for IP-based next generation networks	George Pavlou
INRIA	Abdelkader Lahmadi	Modeling and Performance Evaluation of Networks and Services Management Plane	Olivier Festor, Laurent Andrey
INRIA	Mohamed Nassar	Security of Voice services over Internet - new generation	Olivier Festor, Radu State
INRIA	R�mi Badonnel	Management of Ad-hoc Networks and Services	Radu State, Andr� Schaff
IC	Sye Loong Keoh	A Policy-based Security Framework for Ad-hoc Networks	Emil Lupu
IC	Nilufer Tuptuk	Adaptive Security Risk Management	Emil Lupu
IC	Alberto Egon Schaeffer-Filho	Towards Supporting Interactions between Self-Managed Cells	Emil Lupu
IC	Changyu Dong	Trust and Privacy Management	Naranker Dulay
IC	Driss Choujaa	Learning Techniques for Ubiquitous Computing	Naranker Dulay
IC	Eskindir Asmare	Self-management framework for Unmanned Autonomous Vehicles	Morris Sloman

Institution	PhD Student	Title	Supervisor
HIO	Siri Fagerness	System Administration and Emergent Properties of Pervasive Computing Environments	Mark Burgess
LMU	Feng Liu	Modeling and Planning IT Service Recovery	Heinz-Gerd Hegering
LMU	Ralf König	A tool-kit for the analysis and design of systems with self-management capabilities	Heinz-Gerd Hegering
LMU	Thomas Schaaf	Automation in IT Service Management	Heinz-Gerd Hegering
LMU	Michael Brenner	Tool Support for ITIL-oriented Service Management – A Model-based Approach	Heinz-Gerd Hegering
LMU	Vitalian Danciu	Application of Policy-based Techniques to Process-oriented IT Service Management	Heinz-Gerd Hegering
LMU	Gentschen Felde	Coupling of Security Mechanisms in Grid Environments	Heinz-Gerd Hegering
LMU	Martin Sailer	Concept of a Service MIB – Analysis and Specification of Service-oriented Management Information	Heinz-Gerd Hegering
LMU	Michael Schiffers	Management of Dynamic Virtual Organisations in Grids	Heinz-Gerd Hegering

Most of the PhD students in the list provided an overview of their PhD research work at the AIMS 2007 conference. Those presentations were then discussed with the audience including also most of the supervisors of the EMANICS institutes.

It should be noted that not all descriptions of PhD thesis work were accepted (if they did not satisfy quality requirements as imposed by the AIMS PhD technical committee). The descriptions of PhD works that have been accepted have been included in the AIMS 2007 conference proceedings. However, since the AIMS 2007 proceedings are under copyright of Springer LNCS, these descriptions could not be included in D1.3.¹ Also, further presentations of EMANICS PhD students will take place at the end of July at the EUNICE 2007 Summer School at the University of Twente. Additional education of and discussion amongst PhD students will take place at the First International Summer School on Network and Service Management 2007, organized by the Jacobs University Bremen.

3.2 Common Course Program

Based on the results reported in Deliverable 1.2, work towards a common course program has been continued in the current reporting period. It was accompanied by various discussions how to achieve this ambitious objective due to the fact that it e.g.

- should be applicable for PhD and MSc level and it

¹ Proceedings of AIMS 2007, DSOM 2006, IM 2007, EUNICE 2007 will be provided to the reviewers.

- should allow to exchange students among EMANICS partners by allowing also to accredit credit points of a module that a MSc student has heard at a partner institution.

Related work in defining approaches to do education in IT management such as the ACM Curricula Recommendations (<http://www1.acm.org/education/curricula.html>), ACM Guidelines for Associate-Degree Programs To Support Computing in a Networked Environment (<http://www.acm.org/education/curricula.html#network-env>), and IEEE Educational Activities Board has been studied.

Discussions within work package 1 lead to the question whether the proposed module structure of the common course program is the most appropriate form to achieve our objective to foster exchange of students (PhD and MSc), and to achieve a common level of knowledge among partners. An alternative would be to build a kind of “graduate program” in the area of integrated network and service management, including not only IT management related topics but also basics in networking, distributed algorithms, operating system concepts, cryptographic algorithms, logic and formal methods etc. Work on this will continue in phase 2.

As the first step we started by building an EMANICS teaching material repository by collecting teaching material from EMANICS partners

- on various content related to network and service management, but also topics on networking etc.,
- in various forms (slides, commented slides, labs, podcasts, etc.),
- various languages, and

by adding some meta data to describe the content, objectives, form etc. The repository will be accessible by EMANICS partners over the EMANICS web server, and allows to search for available teaching material. Partners are encouraged to submit new and update existing material. The meta data allows for more efficient search capabilities.

The work package 1 web site contains also a collection of relevant literature in the area of network and service management (not exhaustive), which is as follows:

- *The Practice of System and Network Administration*, by Thomas A. Limoncelli (Author), Christine Hogan (Author), Addison-Wesley Professional; 1ST edition (August 14, 2001)
- *Analytical Network and System Administration: Managing Human-Computer Systems*, by Mark Burgess (Author), Publisher: Wiley (May 28, 2004)
- *Principles of Network and System Administration*, by Mark Burgess (Author), Publisher: Wiley (May 28, 2004)
- *Network and Distributed Systems Management*, by Morris Sloman (Editor), Addison Wesley Longman; 1st edition (June 30, 1994)
- *Integrated Management of Networked Systems: Concepts, Architectures, and Their Operational Application (The Morgan Kaufmann Series in Networking)*, by Heinz-Gerd Hegering (Author), Bernhard Neumair (Author), Sebastian Abeck (Author), Morgan Kaufmann Publishers Inc.; 11th edition (September 26, 2006)

3.3 EMANICS Grants Reflecting Integration and Mobility Support

As reported in Deliverable D1.2, we developed EMIN, a Java-based tool to visualize and document the integration within the NoE. An objective during the tool development was to foresee possibilities of integration with other data collecting tools like the application grant system (where application for mobility grants, joint publications and PhD committees etc. have to be requested) which is first approached. Data of grant requests, as submitted over the EMANICS application grant system, can be visualized in the EMIN integration graphs. The table below gives an overview of the requested grants in the period from April to June 2007. Since April 2007, the new EMANICS grant request system is in place.

Table 2: EMANICS Grants Approved from April to June 2007

Grant Request Type	Participant (Applicant)	Partner(s)	Event
Book Publication grant	Mark Burgess	HIO	Proceedings of the first AIMS EMANICS conference, ISBN 978-3-540-72985
Book Publication grant	Burkhard Stiller	UniZH	Ubiquitous Computing
Conference & Summer School attendance grant	Iris Hochstatter	CETIM	AIMS 2007, Oslo, Norway
Conference & Summer School attendance grant	Gabi Dreo	CETIM	EMANICS General Assembly, AIMS 2007, Oslo, Norway
Conference & Summer School attendance grant	Samah Bel Haj Saad	CETIM	EMANICS General Assembly, AIMS 2007, Oslo, Norway
Conference & Summer School attendance grant	Iris Hochstatter	CETIM	EUNICE 2007 Summer School, University of Twente, the Netherlands
Conference & Summer School attendance grant	Frank Eyermann	CETIM	The Second International Conference on Internet Monitoring and Protection
Conference & Summer School attendance grant	Matthias Goehner	CETIM	WP8 Meeting, Zurich
Conference & Summer School attendance grant	Iris Hochstatter	CETIM	WP8 Meeting, Zurich
Conference & Summer School attendance grant	Nilufer Tuptuk	IC	AIMS 2007, Oslo, Norway
Conference & Summer School attendance grant	Eskindir Asmare	IC	AIMS 2007, Oslo, Norway
Conference & Summer School attendance grant	Radu State (Olivier Festor)	INRIA	AIMS 2007, Oslo, Norway
Conference & Summer School attendance grant	Olivier Festor	INRIA	AIMS 2007, Oslo, Norway
Conference & Summer School attendance grant	Jérôme François	INRIA	AIMS 2007, Oslo, Norway
Conference & Summer School attendance grant	Mohamed Nassar	INRIA	AIMS 2007, Oslo, Norway
Conference & Summer School attendance grant	Abdelkader Lahmadi	INRIA	AIMS 2007, Oslo, Norway
Conference & Summer School attendance grant	Laurent Andrey (Olivier Festor)	INRIA	EMANICS Summer School Bremen

Grant Request Type	Participant (Applicant)	Partner(s)	Event
Conference & Summer School attendance grant	Olivier Festor	INRIA	EMANICS Summer School Bremen
Conference & Summer School attendance grant	Mohamed Nassar (Olivier Festor)	INRIA	EMANICS Summer School Bremen
Conference & Summer School attendance grant	Humberto Abdelnur (Olivier Festor)	INRIA	EMANICS Summer School Bremen
Conference & Summer School attendance grant	Christi Popi (Olivier Festor)	INRIA	EMANICS Summer School Bremen
Conference & Summer School attendance grant	Abdelkader Lahmadi	INRIA	EMANICS Summer School Bremen
Conference & Summer School attendance grant	Radu State (Olivier Festor)	INRIA	EMANICS Summer School Bremen
Conference & Summer School attendance grant	Mohamed Nassar	INRIA	WP2 Meeting participation, Munich
Conference & Summer School attendance grant	Abdelkader Lahmadi (Olivier Festor)	INRIA	WP7 Meeting Participation, Bremen
Conference & Summer School attendance grant	Radu State (Olivier Festor)	INRIA	WP7 Meeting Zurich
Conference & Summer School attendance grant	Juergen Schoen-waelder	IUB	AIMS 2007, Oslo, Norway
Conference & Summer School attendance grant	Iyad Tumar	IUB	AIMS 2007, Oslo, Norway
Conference & Summer School attendance grant	Ha Manh Tran	IUB	AIMS 2007, Oslo, Norway
Conference & Summer School attendance grant	Juergen Schoen-waelder	IUB	EUNICE 2007 Summer School, University of Twente, the Netherlands
Conference & Summer School attendance grant	Ha Manh Tran	IUB	EUNICE 2007 Summer School, University of Twente, the Netherlands
Conference & Summer School attendance grant	Juergen Schoen-waelder	IUB	IM 2007, Munich, Germany
Conference & Summer School attendance grant	Rolf Stadler	KTH	AIMS 2007, Oslo, Norway
Conference & Summer School attendance grant	Fethi Wuhib (Rolf Stadler)	KTH	IM 2007, Munich, Germany
Conference & Summer School attendance grant	Rolf Stadler	KTH	IM 2007, Munich, Germany
Conference & Summer School attendance grant	Ralf Koenig	LMU	AIMS 2007, Oslo, Norway
Conference & Summer School attendance grant	Feng Liu	LMU	EMANICS General Assembly, Oslo, Norway
Conference & Summer School attendance grant	Ralf König	LMU	London General Assembly
Conference & Summer School attendance grant	Oscar Fredy Gonzalez Duque	UniS	AIMS 2007, Oslo, Norway
Conference & Summer School attendance grant	Antonis Hadjiantonis	UniS	AIMS 2007, Oslo, Norway
Conference & Summer School attendance grant	Aimilios Chourmouziades	UniS	EUNICE 2007 Summer School, University of Twente, the Netherlands
Conference & Summer School attendance grant	Aimilios Chourmouziades	UniS	IM 2007, Munich, Germany
Conference & Summer School attendance grant	Burkhard Stiller	UniZH	AIMS 2007, Oslo, Norway
Conference & Summer School	Cristian Morariu (Burk-	UniZH	AIMS 2007, Oslo, Norway

Grant Request Type	Participant (Applicant)	Partner(s)	Event
attendance grant	hard Stiller)		
Conference & Summer School attendance grant	Thomas Bocek (Burkhard Stiller)	UniZH	AIMS 2007, Oslo, Norway
Conference & Summer School attendance grant	Burkhard Stiller	UniZH	AIMS 2007, Oslo, Norway
Conference & Summer School attendance grant	David Hausheer	UniZH	AIMS 2007, Oslo, Norway, Tutorial Chair
Conference & Summer School attendance grant	Burkhard Stiller, P. Kurtansky	UniZH	IM 2007, Munich, Germany
Conference & Summer School attendance grant	Edgar Magana (Joan Serrat)	UPC	AIMS 2007, Oslo, Norway
Conference & Summer School attendance grant	Martin Serrano (Joan Serrat)	UPC	AIMS 2007, Oslo, Norway
Conference & Summer School attendance grant	Martin Serrano (Joan Serrat)	UPC	IM 2007, Munich, Germany
Conference & Summer School attendance grant	Cristi Stefan	UPI	EMANICS Summer School Bremen
Conference & Summer School attendance grant	Ghita Victor (Luminita State)	UPI	EMANICS summer school, Bremen
Conference & Summer School attendance grant	Claudia Gugiu (Luminita State)	UPI	EMANICS summer school, Bremen
Conference & Summer School attendance grant	Ionut Dinca (Luminita State)	UPI	EMANICS summer school, Bremen
Conference & Summer School attendance grant	Florin Smaranda (Luminita State)	UPI	EMANICS summer school, Bremen
Conference & Summer School attendance grant	Tiago Fioreze	UT	AIMS 2007, Oslo, Norway
Conference & Summer School attendance grant	Anna Sperotto	UT	AIMS 2007, Oslo, Norway
Conference & Summer School attendance grant	Aiko Pras	UT	AIMS 2007, Oslo, Norway
Conference & Summer School attendance grant	Anna Sperotto	UT	EMANICS Summer School Bremen
Conference & Summer School attendance grant	Tiago Fioreze	UT	EMANICS Summer School Bremen
Conference & Summer School attendance grant	Anna Sperotto	UT	IM 2007, Munich, Germany
Conference & Summer School attendance grant	Tiago Fioreze	UT	IM 2007, Munich, Germany
Conference & Summer School attendance grant	Aiko Pras	UT	IM 2007, Munich, Germany
Joint Publication grant	Burkhard Stiller	UniZH	CCGrid 2007 Seventh IEEE International Symposium on Cluster Computing and the Grid — CCGrid 2007
PhD Committee grant	Mark Burgess	HIO - KTH - IUB	Three Disciplines in System Administration
PhD Committee grant	Andreas Hanemann	LMU - CETIM	Automated IT Service Fault Diagnosis Based on Event Correlation Techniques
PhD Committee grant	Martin Sailer	LMU - CETIM	Konzeption einer Service-MIB -- Analyse und Spezifikation dienstorientierter Managementinformation

Grant Request Type	Participant (Applicant)	Partner(s)	Event
PhD Student mobility grant	Tarun Varma (Joan Serrat)	UPC	Task 2.3 of Deliverable D9.2
Short term scientific visit grant	Burkhard Stiller	UniZH, CETIM	University of Federal Armed Forces, Munich
Short term scientific visit grant	Burkhard Stiller	UniZH, UPC	Integration of UPC into WP8
Standardization meeting attendance grant	Juergen Schoen-waelder	IUB	NETCONF SMI Design Team Meeting, London
Standardization meeting attendance grant	Rolf Stadler	KTH	NMRG Meeting, Prague

Table 2 lists grants that have been accredited by WP1 to EMANICS partners for PhD integration or mobility support. The number of mobility grants, in particular conference & summer school attendance grants, considerably increased in the last three months. Also, short term scientific visits have been supported. Prof. Burkhard Stiller visited UPC for a one-day workshop to integrate UPC into work package 8 for the second phase of EMANICS. Work regarding different topics of WP8 was presented from both sides and valuable discussions took place wrt. pricing, protocol design in support of pricing, pricing policies, policy description approaches, pricing and game theory, accounting, and the evaluation of economic incentives. The second grant was awarded to the two research groups of UniZH and CETIM for a two-day joint workshop in April that covered aspects of economic management, service management, and other EMANICS-related areas.

3.4 Towards an Ontology of IT Management

As discussions among EMANICS partners have identified an ontology as an essential fundament of further joint activities, we started discussions on this topic already during phase 1 at the work package meeting in Munich in March although it was planned for phase 2. Ontologies are a means of representing knowledge in a formal and structured way, so that the semantics of exchanged concepts are commonly known between all involved partners.

3.4.1 Why ontology of IT management?

For representing knowledge it is necessary to be able to make statements about entire sets of objects. This helps to structure, classify, model, and represent the concepts and relationships pertaining to some subject matter of interest to some community. The meaning of the terms is specified in some way and to some degree. There are different approaches for this purpose with different levels of semantic specification; for the purpose of defining a vocabulary for network and service management we focus on taxonomies and ontologies.

A taxonomy is a collection of vocabulary terms organized into a hierarchical structure. Each term in a taxonomy is in one or more parent-child relationships to other terms in the taxonomy. The additional meaning expressed by taxonomies depends on the meaning of the hierarchical relationship. In a traditional taxonomy the meaning is generalization/specialization, but there are also taxonomies that use 'part-of', 'instance-of' etc. as relationships.

The term ontology has been used in many different ways: It originates from philosophy where it denotes the study of being or existence as well as the basic categories thereof.

In computer science, it can be seen as a generalization of all of the previously described concepts: An ontology is the attempt to formulate a conceptual schema for describing a domain. For this it provides an “explicit formal specification of the terms in a domain and relationships among them” or “a specification of a conceptualization” as Gruber puts it [1]. In the field of artificial intelligence, the term ontology also usually implies that there is a rich and formal logic-based language for specifying meaning of the terms [2]. Ontologies usually provide not only a hierarchical organization of concepts based on the subsumption relation (often called 'is_a', 'subtype' or 'subclass'), but also other relations, whose semantics specify how one concept is related to another. The most common of the semantic relations other than subsumption is the 'part-of' relation, but the ontology engineer can define arbitrary types of relations.

An ontology with its formalism and concepts defines all possible statements of a knowledge base and their relations. A knowledge base can relate to more than one ontology as long as they base on the same formalism. Logics like first order logic or propositional logic are languages in the first case. With their terms, real world objects and their relations to each other can be described. Thus, they suit very well to represent knowledge in a particular domain such as in our case network and service management.

Description logics are a determinable subset of first order logic and evolved from semantic networks due to pressure to formalize what the network means while retaining the emphasis on taxonomic structure as an organizing principle [3]. The idea is to separate the terminology that is used to describe the world from the actual description of the world using this terminology. Description logics therefore provide a language for specifying ontologies. Description Logics focus on categories and their definitions. They express and reason with complex definitions of objects, classes, and relations among them. However, instead of explicitly creating a concept hierarchy, the taxonomic structure in description logics is inferred from the logical definitions of each concept. This way, the hierarchy naturally evolves with the addition of new concepts, providing flexibility and at the same time robustness.

The major strength of description logics is the trade-off between expressiveness in the language and the computational complexity of certain operations. The semantics are clearly defined and formally founded on first-order logic, but at the same time there is an efficient inference mechanism. Separating the terminology from the actual world description enables a more structured and conceptual approach to knowledge modeling. Models can easily be decomposed and composed. Inferring the taxonomic structure instead of explicitly specifying it provides high flexibility and allows an evolutionary approach. With tools from the Semantic Web initiative, this approach seems to be the most promising.

3.4.2 First Steps

The EMANICS ontology on IT Management (EMOM) should provide us with a common vocabulary for discussions both inside and outside the NoE. EMOM will allow us to better structure and categorize the common course program and relate research activities in network and service management world wide. Conference topics can be more clearly defined and joint work can easily be mapped.

A first step toward EMOM was developed in a combined effort at the WP1 meeting in March in Munich where almost all WP1 partners were present and contributed to the

ontology. A graphical representation of it using a MindMapping tool² can be found on the next page. Regarding the content of the Map shown, this is a first version that will evolve with future discussion both inside the NoE as well as with other projects, standardization bodies and the like. Also regarding the graphical presentation in form of a MindMap, this has to be further developed as ontologies are most useful when expressed in machine-readable formats. As briefly stated above, description logics are a suitable knowledge modeling language for that purpose and in particular the Web Ontology Language (OWL) is a good candidate to be considered in case of EMOM.

² The open source tool FreeMind (<http://freemind.sourceforge.net>) has been used to create the MindMap shown.

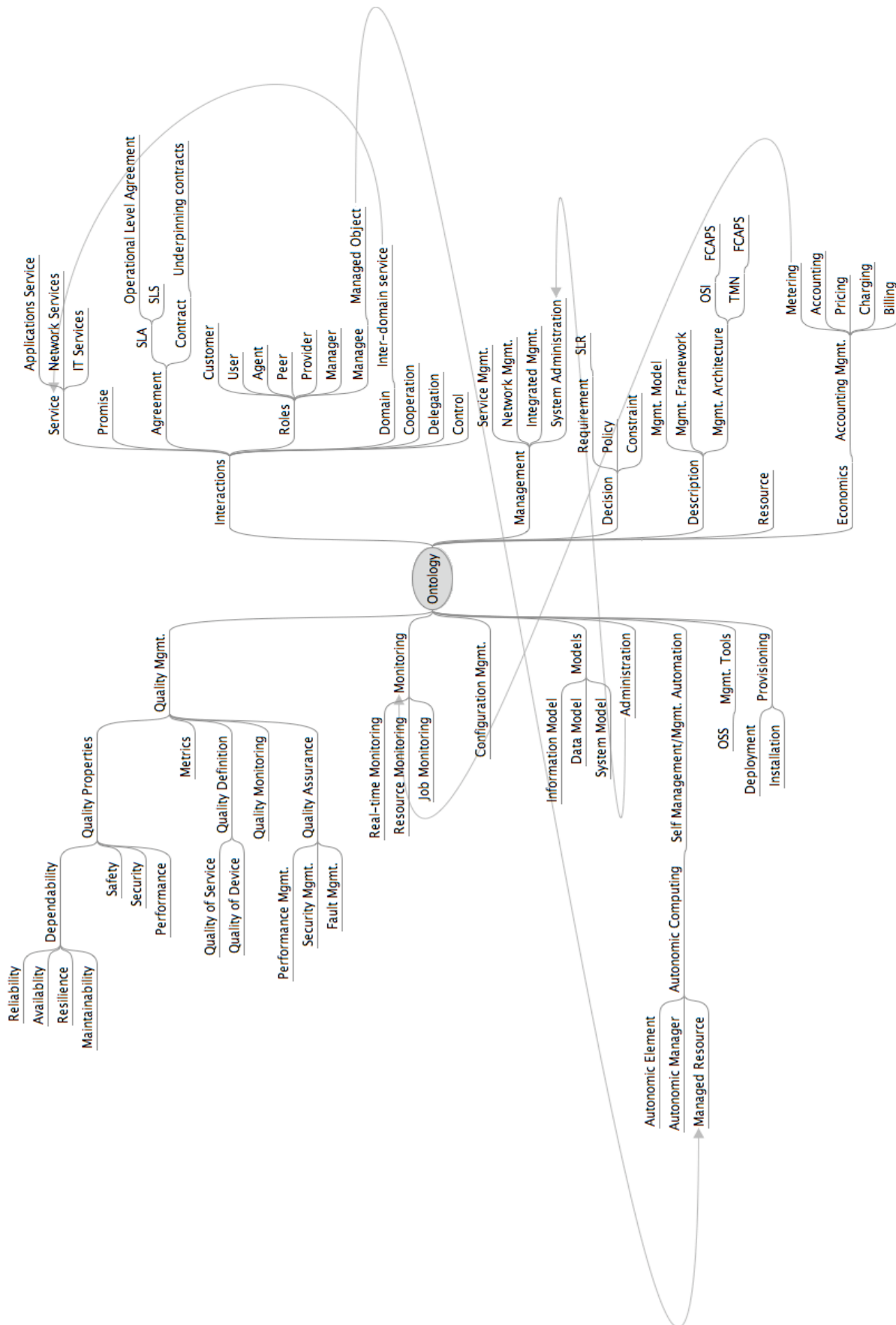


Fig.1 : First achievements towards the ontology

4 Handbook on Network and Service Administration, published by Elsevier

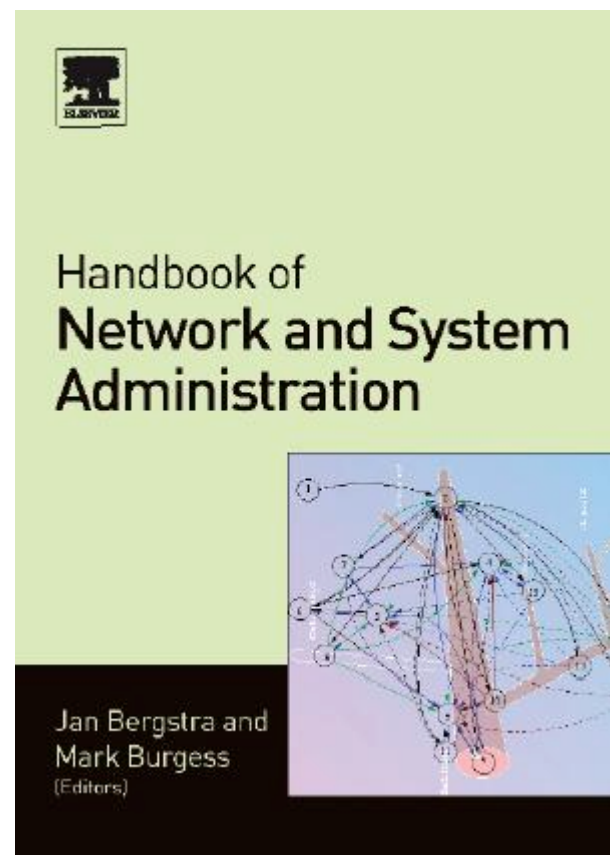
Recently, this handbook has been completed for publication by Elsevier. It is a collection of articles by leading researchers in network and systems administration, many of them partners in the EMANICS Network of Excellence. In this section we will first give an extended table of contents including abstracts for the chapters written by EMANICS partners. The second subsection contains the preface of the handbook by the two editors. All chapters written by EMANICS partners are appended.

Edited by Jan Bergstra and Mark Burgess.

ISBN-13 978-0-444-52198-9

ISBN-10 0-444-52198-4

Publication of this volume is expected in 2007.



4.1 Table of Contents

The handbook presents cutting-edge research results and state-of-the-art in the area of integrated management. The institution in brackets behind the caption lead the effort for the particular chapter. For chapters edited by EMANICS partners an abstract is given.

1. **Introduction** (University of Amsterdam and University College Oslo)
2. **The Arena**
 1. **Commentary** (University of Amsterdam and University College Oslo)

2. Scaling Data Centre Services (University College Oslo)

The management of computer resources is a key part of the operation of IT services. Its trends tend to proceed in cycles of distribution and consolidation. Today, the data centre plays an increasingly important role in services: rather than managing computer resources in small private rooms, many companies and organizations are once again consolidating the management of systems into specialized data centres. Such organized management is linked to the rise of de facto standards like ITIL and to the increased use of outsourcing and service oriented computing which are spreading through the industry. Management standards like ITIL are rather abstract however; this chapter offers a brief technical overview of the data centre environment.

3. Automating System Administration; Landscape, Approaches and Costs (IBM Watson Research Center)

4. System Configuration Management (Tufts University)

3. The Technology

- 1. Editors: Commentary** (University of Amsterdam and University College Oslo)
- 2. Unix and z/OS** (IBM Corporation)
- 3. Email** (University of Amsterdam and TUNIX Internet & Security & Opleidingen)
- 4. XML-based Network Management** (University of Waterloo and POSTECH)
- 5. Open Technology** (TUNIX Internet & Security & Opleidingen and University of Amsterdam)
- 6. System Backup: Methodologies, Algorithms and Efficiency Models** (Exponential Consulting)
- 7. What can Web services bring to Integrated Management?** (University of Twente and NetExpert)

Since the turn of the millennium, Web services have pervaded the middleware industry, despite their technical limitations, their ongoing standardization, the resulting lack of stable standards, the different meanings of the term service to different people, and the fact that marketing forces have blurred the technical reality that hides behind this term. What is so special about Web services and their underlying Service-Oriented Architecture (SOA)?

As a major technology on the software market, Web services deserve to be investigated from an integrated management perspective. What does this middleware technology bring to management applications? Is it yet another way of doing exactly the same thing? Or does it enable manage-

ment software architects to open up uncharted territories for building a new generation of management solutions? Can it help us manage networks, systems and services with unprecedented flexibility, robustness and/or scalability?

In this chapter, we first present the technical motivation for shifting toward Web services. We then describe different facets of this middleware technology: its architecture, its protocols, and the main standards that make up the so-called Web services stack. Furthermore, we show that Web services can be used in integrated management in an evolutionary manner. The example studied here is network monitoring. We propose a fine-grained Web service mapping of SNMP operations and managed objects, and show that the same management tasks can be performed with this new middleware to implement network monitoring. We go back to the rationale behind SOA and compare it with fine-grained Web services. This leads us to propose another perspective, more revolutionary, where services are coarse-grained, wrapping up autonomous programs rather than getting and setting managed objects. We analyze the challenges posed by these coarse-grained Web services to management applications, and the changes that they require in today's management habits. Finally, we give examples showing how management platforms could gradually migrate toward SOA in the future.

8. Internet Management Protocols (Jacobs University Bremen)

Operating a large communication network requires tools to assist in the configuration, monitoring, and troubleshooting of network elements such as switches, routers, or firewalls. In addition, it is necessary to collect event reports to identify and track failures or to provide a log of network activities. Finally, it is necessary to collect measurement data for network planning and billing purposes.

With a handful of devices, many of the activities mentioned above can be supported in an ad-hoc approach by using whatever functionality and interface is provided by a given device. However, when the number of network elements and services increases, it becomes essential to implement more advanced management approaches and deploy specific management tools. Since communication networks often consist of a large number of devices from different vendors, the need for standardized management interfaces becomes apparent.

One particularly important management interface for standardization and interoperability is the interface between network elements and their management systems. Network elements are typically closed systems and it is not possible to install management software on the devices themselves. This makes network elements such as routers and switches fundamentally different from general purpose computing systems. Interfaces between management systems have seen less need for standardization since management systems usually run on full featured host systems and it is therefore easier to use general middleware technologies as a means to provide connectivity between management systems.

This chapter surveys Internet management protocols that are mainly used between network elements and their management systems. Since it is impossible to discuss all protocols in detail, the discussion in this chapter will focus on the requirements that have driven the design of the protocols and the architectural aspects through which the protocols address the requirements. By focusing the discussion on requirements and architectural concepts, it becomes possible to document basic principles that are not restricted to the current set of Internet management protocols, but which will likely apply to future management protocols as well.

4. Networks, Connections and Knowledge

- 1. Commentary** (University of Amsterdam and University College Oslo)
- 2. Management of Ad hoc Networks** (LORIA-INRIA Lorraine MADYNES)

Mobility and the rising number of devices lead to the relatively new paradigm of ad-hoc networks and especially mobile ad-hoc networks (MANET) where devices connect directly to each other. Ad-hoc networks allow for many new services on the fly, but they also pose new requirements on the management by their dynamics, heterogeneity, resource constraints and overall unknown environment. To allow for good connectivity even between devices that cannot reach each other directly, routing mechanisms are of particular interest. The chapter discusses different management organizational models and the main management paradigms for MANETs. This information is completed with the description of different management application domains including monitoring, configuration and co-operation control.

- 3. Some relevant aspects of Network Analysis and Graph Theory** (Telenor R&D)
- 4. Knowledge Engineering using Ontologies** (Motorola Labs)
- 5. Application Integration using Semantic Web Services** (Delft University of Technology)

5. Policy and Constraint

- 1. Editors: Commentary** (University of Amsterdam and University College Oslo)
- 2. Security Management and Policies** (University of California at Davis)
- 3. Policy Based Management** (Imperial College London)

Policies are rules governing the choices in behavior of a system. They are often used as a means of implementing flexible and adaptive systems for management of internet services, distributed systems, and security systems. There is also a need for a common specification of security policy for large-scale, multi-organizational systems where access control is implemented in a variety of heterogeneous components. In this chapter we survey both security and management policy specification approaches.

We also cover the issues relating to detecting and resolving conflicts which can arise in the policies and approaches for refining high level goals and service level agreements into implementable policies. The chapter briefly outlines some of the research issues that have to be solved for large-scale adoption of policy-based systems.

6. Computational Theories of System Administration

1. **Editors: Commentary** (University of Amsterdam and University College Oslo)
2. **On the Complexity of Change and Configuration Management** (University College Oslo)

How hard is it to build and tune a computer system so that one can predict its operational behavior from an original specification?

In computer programming, the boundaries of this problem have been considered at length (see the chapter by Bergstra and Bethke in this volume, for instance). In system administration, the framework for the problem is both larger and potentially much less reliable. System administration is not only a set of procedures executed on a computer, it also involves the interaction of humans in a more pervasive way. It raises questions of a philosophical nature: are we really in control of computer behavior, or are there limitations to what we can achieve? In computer programming, one does not normally take into account the random arrival of input, or the interactions between independent tasks which share common resources. These are matters that are of central importance in system administration.

Suppose we were indeed able to rule or govern the behavior of a computer system, by choosing the values of all the right registers, how hard would it be to find out what values or patterns of data were required to map the specification of those values to a desired behavior? How much effort would we need to put in to solve an arbitrarily posed problem of this kind? What are our chances of being able to optimize it, or simply do it at all?

These are questions we discuss in this chapter. To begin formulating an answer we must cross several bridges, from empirical system administration and more formal computational disciplines such as languages and algorithmic complexity. We discuss what it means to talk about the difficulty of arranging a particular computer configuration and why we believe the choice of configuration has something to do with its ultimate behavior.

There are many measures one might use to discuss the complexity of a solution to a problem: the number of lines of code required to solve the problem, the number of hours of planning needed, the CPU cycles expended during the task, the memory resources used to run the algorithms, etc. All of these are potential measures of difficulty or complexity, but to answer the question in a comprehensible way, we have to pose the question in a way that can be analyzed.

None of the measures above have any meaning to system administration until we can identify a set of acceptable tests and determine the answer to a question: does the solution pass the test? Yes or no? So, how shall we proceed? Our story is a complicated one; it requires both a broad background and lengthy argumentation. To make it comprehensible to the widest possible audience, we have chosen to use a pedagogical style with a moderate amount of introductory background. We hope that this makes our discussion as self-contained as possible. Perhaps half of the material will be known to about half our audience, but we doubt that these halves will overlap.

We base our discussion on classical computational complexity theory and the modern theory of configuration management in terms of operations. Many readers will have heard of phrases like “np complete” used to describe hard problems, and understand that therein lie demons and dragons concerning the difficulty of a task. But what does this really mean? Similarly, readers will have heard about the problem of configuration management and computer maintenance. Why is this any different from computer programming? We begin from the context of computer management.

3. Complexity of System Configuration Management (Tufts University)

4. Predictable and Reliable Program Code (University of Amsterdam)

7. System Science

1. Editors: Commentary (University of Amsterdam and University College Oslo)

2. System Administration and the Scientific Method (University College Oslo)

Networks and computer systems are objects to be observed and understood by a process of scientific inquiry. Even in isolation, computers are too complicated to predict from purely deterministic principles of logic and reason; then, once we place them in a non-deterministic environment of users and networks, the task becomes impossible. The result is that network or system manager is in the position of the natural scientist observing un-known phenomena and learning about behavior that he or she strives to explain.

Computer Science looks traditionally to logic and mathematics rather than to natural philosophy to describe computers, examining the consistency of assumptions rather than the reality of their implementation. Clearly such studies are valuable, but they do not represent the practice of computer systems. Scientists have long since abandoned such fundamental approaches to studying the real world. Trying to describe the behavior of the human body in terms of atoms and their basic interactions, for instance, is simply not a tractable problem.

Science has learned to eat humble pie and make do with suitably idealized approximation. To defend itself from unreasonable criticism, it has then

developed a theory for gauging the uncertainty of the conclusions it reaches. In the engineering world, this scientific approach is not only useful but essential. We can distinguish two complementary approaches to understanding the world. Put very simply:

- Empiricism: to observe, model and explain.
- Mathematics: to assume, formulate and deduce.

Science, or natural philosophy, is the discourse between observation and rational approximation. It is about finding a suitably idealized description of the world by which we might summarize its essence and use the information to good effect. It begins with empiricism, or observation. Mathematics and logic, on the other hand, are precise constructs based on rewriting assumptions according to unambiguous rules. What goes in, comes out in a new and enlightening form.

There are two reasons for modeling: the first is to make sure that one's assumptions actually lead to the outcomes one imagines, through a plausible chain of cause and effect, and also to within an acceptable margin of error. The second is to make predictions about new scenarios that we have never seen, based on past observation and modeling. The world, of course, never obeys simple laws and models exactly: all models are 'wrong', in a sense, but that need not matter as long as we know how wrong they are likely to be.

System administration is about the planning, deployment and maintenance of human-computer systems. The human element of the system here, since it leads to significant uncertainty. Humans make approximation essential.

3. System Administration and Business (University College Oslo)

The integration of computer technology into a business process is an obvious goal for an organization or company that relies on information services. In this chapter we ask: is it possible to understand business modelling and analytical system administration in the same light? In other words, how do we bring a rational scientific method to bear on the problem of building a business around an Information Technology (IT) infrastructure, or building an IT infrastructure for an existing business?

To do this, we must view process of business modelling in the same light as the process of modelling the rest of the information system, and to relate these ideas to the practical issues that businesses have to contend with.

The human component of both businesses and human-computer systems requires us to consider both rational decision making (optimization based on objective criteria) and irrational choices (arbitrary choices made as policy for reasons that are not part of a scientific framework, e.g. ethics or image). For example:

- Standards of "best practice". Why are they best?

- Modeling a business process as an information system.
- Logistics of production and inventory.
- Identification of relevant scales of operation.
- Capacity planning.
- Time management.
- Strategic thinking: activity patterns like type I models, and decision planning like type II models.
- Gauging risks and uncertainties.
- Minimizing risk occurrences.
- Finding policies as stable equilibria.

One strategy for integrating business thinking into network and system management is to view activities in terms of services, rather than as low-level protocol interactions. This has become a popular paradigm since the ITIL/BS15000 language of service provision and Service Level Agreements etc. See also the work in the Telemanagement Forum with eTOM and DEN-ng.

Although the individual details of procedures are important, there is a more overarching need to understand the logistics of combining such procedures in the environment of an economic motor. We wish to turn some of these loose ideas into more rigorous ideas so that one can apply the techniques of scientific modeling and rational decision making to them.

Not all models that are related to aspects of a business are 'business models' in the fuller sense of the word. There are good reasons to separate off specialized issues and consider them separately. A business model claims to supply the *raison d'être* of a business, along with its capacity for long term profitability. Like a computer system, a business is a box surrounded by an environment with which it is interacting.

Modeling something as complex as a business in full is probably not a practical proposition, using today's technology. Perhaps in the future one might model businesses as one today models the weather, but we are some way off having this technology today.

A key difference with computer management is that businesses tend to seek growth rather than stability.

We shall also look at inventory models, efficiency models and so on, which pertain to smaller and more specialized parts of the whole.

The plan for this article is:

- Modeling business as a human-computer system.

- Identifying the principle for optimization.
- Managing internals, such as inventory and configuration.
- Planning high volume services and controlling uncertainties.

4. System Reliability (University of Oslo)

8. Business and Services

1. **Editors: Commentary** (University of Amsterdam and University College Oslo)
2. **State of the Art in Economics Management of Internet Services** (University of Zurich)

Managing the heterogeneous and partly broadband-capable infrastructure of tomorrow's Internet requires suitable and scalable technology as well as mechanisms in support of viable economic means of service deployment and provisioning. Such services will advance beyond the pure Internet Protocol (IP) access and need to cover value-added services as well. Currently, an economic unfairness exists in the current Internet and in a converged IP infrastructure also. This state-of-the-art in traditional network management and related basic economics focuses mainly on the use case of network service provisioning. Newly developed, integrated concepts of network management and economics, and ideas of emerging, urgent, and required research questions, technology requirements, and future economic management schemes have been identified and discussed briefly. Driven by these considerations, the need for developing advanced network management techniques becomes obvious, of course, only based on well defined roles and models, covering promises and interactions, that will enable multiple parties to access this infrastructure in a technically suitable, efficient, and economically fair manner. This state-of-the-art work, which goes partly even one step beyond the current state of network management — in terms of models and mechanisms — has been termed EMIS (Economic Management of Internet Services). This approach shall lead to a common and administrable information infrastructure, which in turn can be operated economically and efficiently under predefined assumptions.

Note: Deliverable 8.1 was of such excellent quality that it made its way to the handbook as chapter 8.2 after some minor editorial changes.

3. **Service Provisioning: Challenges, Process Alignment and Tool Support** (Ludwig Maximilians University Munich and University of Federal Armed Forces Munich and Leibniz Supercomputing Center)

The provisioning of services in the field of information and communication technology (ICT) such as network and application services presents many challenges to service providers. Service providers must pay careful attention to operational efficiency and take new approaches to service provisioning in order to compete successfully. They must make better use of existing resource infrastructures, such as networks and servers, as well as

reduce the number of operational staff required to deliver services to end users leveraging automation.

Regarding the level of service customization, services span a broad range: from widely deployed commodity services, such as broadband internet access or e-mail services, with millions of equal service instances, to individual services, such as an enterprise extranet or an in-house service like the service architecture of a world wide web (WWW) search engine with a single highly customized service instance.

The vision of a service provider and also a competitive advantage is the achievement of “flow-through provisioning”, which means that a customer reviews a set of services offered to him by a provider, selects an appropriate service, chooses among the available service quality options, then places an accordingly formed service order, and waits for its almost instantaneous fulfillment. The service provider, who receives such an order, would have automated provisioning systems that activate, monitor and manage the ordered services with minimal additional manual input. So far, this has partly been accomplished for a few types of commodity services, but it remains a vision for most services that require a higher level of customization.

4. **IT Service Management** (Verdonck, Klooster and Associates and Motorola Labs)
5. **Decision and Control Factors for IT-sourcing** (Verdonck, Klooster and Associates)
6. **How do ICT Professionals Perceive Outsourcing? Factors that influence the success of ICT outsourcing** (Verdonck, Klooster and Associates)

9. Professional and Social Issues

1. **Commentary** (University of Amsterdam and University College Oslo)
2. **Systems Administration as a Self-Organizing System: the Professionalization of SA via Interest & Advocacy Groups** (VirtualNet Consulting)
3. **Ethical, Legal and Social Aspects of Systems** (University College Oslo)

Communication networks have altered many aspects of life, like work and employment, healthcare, transportation and entertainment. Technology has brought many benefits, but is also the cause of many social and ethical concerns: system failures, the destruction of property, the so called digital divide, workplace monitoring, Internet censorship, and the increasing gap between rich and poor, are some examples of important ethical questions that arise when we reflect on how technology has impacted modern life.

System Administrators play an important part in keeping networks secure. They are also responsible for the uninterrupted operation of the computers

to take care of the business needs. A system administrator's duties and responsibilities encompass more than technical knowledge. They are given broad access to the resources of computer systems because their job responsibilities require such access. For instance, in troubleshooting email issues, a system administrator may have to go through the employees' email, but must treat any knowledge so gained as private. Many companies have codes of ethics for employees who in the nature of their job have privileged access to company systems. The content of such codes may include statements like: 'do not browse through the computer information of system users while using the powers of privileged access unless such browsing: is a specific part of the job description, do not disclose computer information observed while operating with privileged access, do not copy any computer information for any purpose other than those authorized under their defined job responsibilities, report suspicious incidents to management or police' and so on. Such codes of ethics provide guidelines for professional conduct, and are useful tools. However, situations do arise where it may be hard to decide what to do, for instance, how would you react if your boss asks you to go through all the employees' email?

4.2 Preface

To give some details about the contents of the handbook, we cite the preface by Mark Burgess and Jan Bergstra:

"The volume, of which you are in some doubt could reasonably be called a book to be held in human hands, is a collection of chapters sampling the current State of the Art of network and system administration or management. It is written by researchers, for researchers, educators and advanced practitioners.

The field of Network and System Administration is (some would claim) still an emerging field, one that bridges disciplines, and challenges the traditional boundaries of computer science. It forces us to confront issues like partial predictability and partial reliability that are more commonly associated with engineering disciplines in an uncertain environment. It does so because human-computer networks are complex and uncertain environments. In other words, it rediscovers the lesson that other sciences have already made peace with: that approximation is the only realistic approach to its difficulties.

Whatever the novelty of the field, there is already a large body of knowledge about it, contained in a diverse array of sources. This has only recently begun to be formalized however, in the way we conventionally expect of an academic discipline. Much of it has been only sketchily formulated and does not lend itself to future development by researchers and designers. We hope to partially rectify that situation with this volume.

So what is administration or management? This is a difficult question to answer in full, but it clearly covers few themes:

- Making choices and decisions about design, implementation, practice and operations.
- Provision of resources.
- Infrastructure and service design.
- Maintenance and change management.

- Verification and validation of specifications.
- Economic usage and business integration.

Within the list above, we also find the foundations for more abstract concerns like security and commerce. We have not been able to cover every aspect of the field in this book, but we have hopefully covered enough to take us forward.

Our Handbook of Network and System Administration is not a textbook for day-to-day practitioners. It is aimed at those who are seeking to move beyond recipes and principles, on to creative innovation in application, development or research; it offers a solid scientific and engineering foundation for the field. We hope that it will also inspire less experienced readers. For them we recommend a background for this volume in a number of recent textbooks:

- For a technical introduction to the principles and techniques of system administration one has Principles of Network and System Administration, Mark Burgess. Wiley (2000, 2004).
- An introduction to the heuristics and day to day practices of system administration is found in The Practice of System and Network Administration, T.A. Limoncelli and C. Hogan. Addison Wesley (2002)
- Advanced matters of modeling and in-depth understanding of systems can be found in Analytical Network and System Administration, Mark Burgess. Wiley (2003).

In addition to this educationally honed sequence, there are numerous resources on specific technologies, for instance in the 'animal series' of books by the publisher O'Reilly.

If we add to the above a book of collected papers, which documents the early work in system administration,

- Selected Papers in Network and System Administration, Erik Anderson, Mark Burgess and Alva Couch Eds., Wiley (2001)

and a book of chapters on the early parts of Network Management

- Network and Distributed Systems Management, Morris Sloman Ed., Addison Wesley (1994),

then we could present this volume as the next piece of a series covering the academic spectrum in the field.

An editor quickly discovers that soliciting contributions is harder than one might think. Shamelessly appealing to the vanity of contributors only goes so far in coaxing words onto pages. Those who have practiced network and system management, in particular, rarely like to write down their knowledge.

This has made our volume end up being more forward-looking than historical, although we have succeeded in covering a few examples of classic materials, such as E-mail and backup. It has left us to try to mention the omissions in our commentary, with obvious limitations. Fortunately many traditional topics are already discussed in the books above.

Our book is already quite thick, so let us not prolong the introduction unnecessarily. We simply invite you to read the chapters and we offer some brief commentary along the way.

Acknowledgements

Jan Bergstra acknowledges support from the NWO-Jacquard grant Symbiosis. The Symbiosis project focuses on IT sourcing in general and software asset sourcing more specifically.

Mark Burgess acknowledges support from the EC IST-EMANICS Network of Excellence (#26854)."

5 Summary and Conclusions

This deliverable presents primarily the combined effort of all EMANICS partners to determine challenges, report on cutting edge research results and identify visions in the field of network and service management in form of the "Handbook on Network and System Administration". The renowned publishing house Elsevier will release the handbook in 2007 and this will increase EMANICS visibility in the community even more.

In addition, we reported on the initial PhD student database, discussion wrt. the common course program and a first step toward an ontology on IT management that will foster a more clearly defined problem space and semantics. This may be used throughout future activities in WP1 in phase II like common course program, worldwide research map, EMANICS integration and collaboration, and a joint PhD program. The results can even be promoted to standardization and regulatory bodies to clarify topics.

Regarding the identification of new challenges, orientations and visions of network and service management, the handbook delivered a current status, but work on this topic will be a major topic in the second phase of the NoE. The next event planned will take place already at the end of July 2007 at Dagstuhl, the seminar "Autonomic Management of Networks and Services" (<http://www.dagstuhl.de/de/programm/kalender/semhp/?seminr=2007302>).

6 References

- [1] Gruber, Thomas R.: A translation approach to portable ontology specifications. Knowledge Acquisition, 5(2): pp. 199-220, 1993.
- [2] Uschold, Michael, Grüninger, Michael: Ontologies: principles, methods, and applications. Knowledge Engineering Review, 11(2): pp. 93–155, 1996.
- [3] Baader, Franz, Calvanese, Diego, McGuinness, Deborah, Nardi, Daniele, Patel-Schneider, Peter F. (editors): The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press, 2003.

7 Acknowledgement

This deliverable was made possible due to the large and open help of the WP1 team of the EMANICS NoE. Many thanks to all of them.

8 EMANICS contributions to the “Handbook of Network and System Administration”

As stated in section 3, chapters edited by EMANICS partners are part of this deliverable and follow in the order of their appearance in the book:

Chapter 2.2 Scaling Data Centre Services (University College Oslo)

Chapter 3.7 What can Web services bring to Integrated Management? (University of Twente and NetExpert)

Chapter 3.8 Internet Management Protocols (Jacobs University Bremen)

Chapter 4.2 Management of Ad hoc Networks (LORIA-INRIA Lorraine MADYNES)

Chapter 5.3 Policy Based Management (Imperial College London)

Chapter 6.2 On the Complexity of Change and Configuration Management (University College Oslo)

Chapter 7.2 System Administration and the Scientific Method (University College Oslo)

Chapter 7.3 System Administration and Business (University College Oslo)

Chapter 8.2 State of the Art in Economics Management of Internet Services (University of Zurich)

Note: As this chapter is already known as Deliverable 8.1, we do not append it to this deliverable again; only minor adjustments have been made to the introduction.

Chapter 8.3 Service Provisioning: Challenges, Process Alignment and Tool Support (Ludwig Maximilians University Munich and University of Federal Armed Forces Munich and Leibniz Supercomputing Center)

Chapter 9.3 Ethical, Legal and Social Aspects of Systems (University College Oslo)

Scaling Data Centre Services

Mark Burgess

November 29, 2006

1 Introduction

The management of computer resources is a key part of the operation of IT services. Its trends tend to proceed in cycles of distribution and consolidation. Today, the data centre plays an increasingly important role in services: rather than managing computer resources in small private rooms, many companies and organizations are once again consolidating the management of systems into specialized data centres. Such organized management is linked to the rise of de facto standards like ITIL[26, 25, 4] and to the increased use of outsourcing and service oriented computing which are spreading through the industry. Management standards like ITIL are rather abstract however; this chapter offers a brief technical overview of the data centre environment.

2 Computing services

Computing services are increasingly required to be scalable and efficient to meet levels of demand. The delivery of high levels of computing service is associated with three common phrases:

- High Performance Computing (HPC).
- High Volume Services (HVS).
- High Availability Services (HAS).

These phrases have subtly different meanings.

2.1 High Performance Computing

HPC is the term used to describe a variety of CPU intensive problems, This often involves supercomputers and/or networked clusters of computers to take a large computational problem, divide it into smaller chunks and solve those chunks in parallel. HPC requires very high-capacity and/or low-latency connections between processors. The kinds of problems solved in this way are typically of a scientific nature, such as massive numerical problems (e.g. weather and climate modelling), or search problems in bioinformatics.

For problems that require massive computational power but which are less time-critical, another strategy is *Grid computing*, an extension of the idea in which the sharing takes place over anything from a short cable to a wide area network.

2.2 High Volume Services

This refers to applications or services, normally based on content provision, that are designed to handle large numbers of transactions. HVS are a generalization of HPC to non-specifically CPU oriented tasks. High volume services are achieved by implementing strategies for workload sharing between separate server-hosts (often called load balancing).

HVS focuses on serving large volumes of requests, and must therefore address scalability. System throughput demands a careful analysis of system bottlenecks. Load-sharing is often required to attain this goal. HVS includes the economics and the performance tuning of services. The ability to deliver reliable service levels depends on both the resources that are available in the data centre and the pattern of demand driven by the users.

2.3 High Availability Services

These are applications designed for *mission-critical* situations, where it is essential that the service is available for the clients with a *low latency*, or short reponse time. To achieve this, a mixture of redundancy and fault detection is implemented in the system. HAS is about ensuring that a service is available with an assured quality of response time, with target service level. One must decide what “available” means in terms of metric goals (e.g. upper bound response time).

3 Resources

To achieve a high performance (and hence high volume) computing we need resource management. The resources a computer system has to utilize and share between different tasks, users and processes are

- CPU.
- Disk.
- Memory.
- Network capacity¹.

Many technologies are available to do optimization and perform this sharing, as noted in the forthcoming sections.

Computer processing is limited by *bottlenecks* or throughput limitations in a system. Sometimes these performance issues can be separated and conquered one by one, other times interdependencies make it impossible to increase performance simply by making changes to components. For instance, some properies of systems are *emergent*, i.e. they are properties of the whole system of all components, not of any individual part.

3.1 CPU

For CPU intensive jobs, the basic sharing tools are:

- Multi-tasking and threading at the operating system level.
- Multi-processor computers, vector machines and Beowulf clusters which allow parallelism.

¹The use of the term “bandwidth” is commonly, if incorrectly, used in everyday speech. Bandwidth refers to frequency ranges. Channel capacity (or *maximum throughput*) is proportional to bandwidth – see chapter System Administration and the Scientific Method in this volume.

- Mainframe computing.
- Grid-Engine job distribution systems for parallel computing.
- MPI (Message Passing Interface) programming tools

Flynn’s taxonomy is a classification scheme for parallel computers based on whether the parallelism is exhibited in the instruction stream and/or in the data stream. This classification results in four main categories, SISD (single instruction single data), SIMD (single instruction multiple data), MISD (multiple instruction single data) and MIMD (multiple instruction multiple data). Different strategies for building computers lead to designs suited to particular classes of problems.

Load-balancing clusters operate by having all workload come through one or more load-balancing front ends, which then distribute work to a collection of back end servers. Although they are primarily implemented for improved performance, they commonly include high-availability features as well. Such a cluster of computers is sometimes referred to as a server farm.

Traditionally computers are built over short distance buses with high speed communications. Dramatic improvements in network speeds have allowed wide area connection of computer components. Grid computing is the use of multiple computing nodes connecting by such wide area networks. Many grid projects exist today; many of these are designed to serve the needs of scientific projects requiring huge computational resources, e.g. bioinformatics, climate and weather modelling, and the famous Search for Extra-terrestrial Intelligence (SETI) analysis project.

3.2 Disk

Fast, high volume storage is a key component of delivery in content based services. Because disk-storage requires mechanical motion of armatures and disk heads, reading from and writing to disk is a time-consuming business, and a potential bottleneck. Disk services that cache pages in RAM can provide a large speed-up for read/write operations.

RAID includes a number of strategies for disk performance and reliability. Redundant Arrays of Independent Disks² are disk arrays that have special controllers designed to optimize speed and reliability. RAID deals with two separate issues: fault tolerance and parallelism. Fault tolerant features attempt to reduce the risk of disk failure and data-loss, without down-time. They do this using redundant data encoding (parity correction codes) or mirroring. Parallism (mirroring and striping of disks) is used to increase data availability. These are often conflicting goals[15, 16, 11, 31].

Disk striping, or parallelization of data access across parallel disks can result in an up-to- N -fold increase in disk throughput for N disks in the best case.

Error correction, sometimes referred to as *parity*, is about reliability. Error correction technology adds additional search and verification overheads, sometimes quoted at as much as 20% to the disk search response times. It can be a strategy for increasing up-time however, since RAID disks can be made hot-swappable without the need for backup.

Ten years ago networks were clearly slower than internal databuses. Today network speeds are often faster than internal databuses and there is a potential gain in both performance and administration in making disk access a network service. This has led to two architectures for disk service:

- Storage Area Networks (SAN).

²Originally the “I” meant Inexpensive, but that now seems outdated.

- Network Attached Storage (NAS).

A Storage Area Network is an independent network of storage devices that works in place of disks connected on a local databus. This network nevertheless appears as a low level disk interface, usually SCSI. Disk storage appears as locally attached device in a computer operating system’s device list. The device can be formatted for any high level filesystem.

SAN uses iSCSI (SCSI over Internet) using a dedicated network interface to connect to a storage array. This gives access to an almost unlimited amount of storage, compared to the limited numbers of disks that can be attached by the various SCSI versions. SAN can also use Fibre channel, which is another fibre-based protocol for SCSI connection.

NAS works more like a traditional network service. It appears to the computer as a filesystem service like NFS, Samba, AFS, DFS, etc. This service is mounted in Unix or attached as a logical “drive” in Windows. It runs over a normal network protocol, like IP. Normally this would run over the same network interface as normal traffic.

3.3 Network

Network service provision is a huge topic that evolves faster than the timescale of publishing. Many technologies compete for network service delivery. Two main strategies exist for data delivery:

- Circuit Switching.
- Packet Switching.

These are roughly analogous to rail travel and automobile travel respectively. In circuit switching, one arranges a high speed connection between certain fixed locations with a few main exchanges. In the packet switching, packets can make customized, individually routed journeys at the expense of some loss of efficiency.

Network throughput can be improved by adapting or tuning:

- Choice of transport technology, e.g. Fibre, Ethernet, Infiniband, etc.
- Routing policies, including the strategies above.
- Quality of Service parameters in the underlying transport.

Most technological variations are based on alternative trade-offs, e.g. high speed over short distances, or stable transport over wide-areas. The appropriate technology is often a matter of optimization of performance and cost.

4 Servers and workstations

Computer manufacturers or “vendors” distinguish between computers designed to be “workstations” and computers that are meant to be “servers”. Strictly speaking, a server is a *process* that provides a service not necessarily a whole computer; nevertheless, this appellation has stuck in the industry.

In an ideal world one might be able to buy a computer tailored to specific needs, but mass production constraints lead manufacturers to limit product lines for home, businesses and server rooms. Home computers are designed to be cheap and are tailored for gaming and multi-media. Business computers are more expensive versions of these with different design look. Servers are designed for reliability and have their own “looks” to appeal to data centre customers.

A server computer is often designed to make better use of miniaturization, has a more robust power supply and better chassis space and expansion possibilities. High quality hardware is also often used in server-class systems, e.g. fast, fault tolerant RAM, and components that have been tested to higher specifications. Better power supply technology can lead to power savings in the data centre, especially when many computers are involved in a large farm of machines.

We sometimes think of servers as “large computers”, but this is an old fashioned view. Any computer can now act as a server. Basic architectural low-level redundancy is a selling point for mainframes designs, as the kind of low level redundancy they offer cannot easily be reproduced by clustering several computers.

4.1 Populating a data centre

How many computers does the data centre need, and what kind? This decision cannot be made without an appraisal of cost and performance. In most cases companies and organizations do not have the necessary planning or experience to make an informed judgement at the time they need it, and they have to undergo a process of trial-and-error learning. Industry choices are often motivated by the deals offered by sellers rather than by an engineering appraisal of system requirements, since fully rational decisions are too complicated for most buyers to make.

One decision to be made in choosing hardware for a server function is the following. Should one:

- Buy lots of cheap computers and use high-level load sharing to make it perform?
- Buy expensive computers designed for low-level redundancy and performance?

There is no simple answer as to which of these two approaches is the best strategy. The answer has fluctuated over the years, as technological development and mass production have advanced. Total cost of ownership is usually greater with custom-grown systems and cheap-to-buy computers. The cost decision becomes a question of what resources are most available in a given situation. For a further discussion of this see the chapter on System Administration and Business in this volume.

Computer “blades” are a recent strategic design aimed at data centres, to replace normal chassis computers.

- Blades are small – many of these computers can be fitted into a small spaces specially designed for racks.
- Power distribution to blades is simplified by a common chassis and can be made redundant (though one should often read the small-print).
- A common switch in each chassis allows efficient internal networking.
- VLANs can be set up internally in a chassis, or linked to outside network.
- Blade chassis are increasingly designed for the kind of redundancy that is needed in a large scale computing operation, with two of everything provided as options (power, switch, etc).
- They often have a separate network connectio for an out-of-band management console.

The blade servers are a design convenience, well suited to life in data centre where space, power and cooling are issues.

4.2 Bottlenecks — why won't it go faster?

Service managers are sometimes disappointed that expensive upgrades do not lead to improved performance. This is often because computing power is associated with the CPU frequency statistics. Throwing faster CPUs at service tasks is a naive way of increasing processing. This will only help if the service tasks are primarily CPU bound.

So where are the main performance bottlenecks in computers? This is probably the wrong question to ask. Rather we have to ask, where is the bottleneck in an *application*? The resources needed by different applications are naturally different. We must decide whether a process spends most of its time utilizing

- CPU?
- Network?
- Disk?
- Memory?

We also need to understand what *dependencies* are present in the system which could be a limiting factor[6].

Only when we know how an application interacts with the hardware and operating system can we begin to tune the performance of the system. As always, we have to look at the resources being used and choose a technology which solves the problem we are facing. Technologies have design trade-offs which we have to understand. For instance, we might choose Infiniband for short connection, high-speed links in a cluster, but this would be no good for connecting up a building. Distance and speed are often contrary to one another.

Read versus write performance is a key factor in limiting throughput. Write-bound operations are generally slower than read-only operations because data have to be altered and caching is harder to optimize for writing. Redundancy in subsystems has a performance cost, e.g. in RAID 5 data are written to several places at once, whereas only one of the copies has to be read. This explains why RAID level 5 redundancy slows down disk writing, for example. The combination of disk striping with each disk mirrored is often chosen as the optimal balance between redundancy and efficiency.

Applications that maintain *internal state* over long-term *sessions* versus one-off transactions often have complications that make it harder to share load using high-level redundancy. In such cases low-level redundancy, such as that used in mainframe design is likely to lead to more efficient processing because the cost of sharing state between separate computers is relatively high.

5 Designing an application architecture

5.1 Software architecture

In an ideal world, software engineers would be aware of data centre issues when writing application software. Service design involves concerns from the low level protocols of the network to the high level web experiences of the commercial Internet.

While low level processor design aims for greater and greater integration, the dominant principle in soft-system architecture is the *separation of concerns*. Today's archetypical design for network application services is the so-called *three tier architecture*: webserver, application, database (see fig. 1).

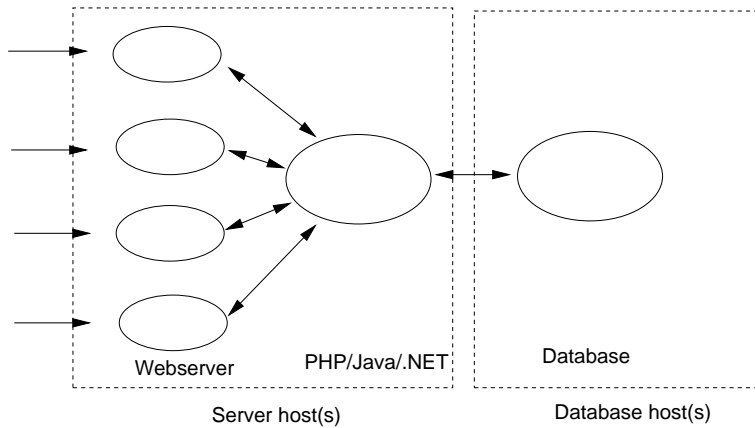


Figure 1: A three tier architecture.

Service design based entirely on the program code would be a meaningless approach in today's networked environments. A properly functioning system has to take into account everything from the user experience to the basic resources that deliver on the service promises. Design considerations for services include:

- Correctness.
- Throughput and scalability.
- Latency (response time).
- Reliability.
- Fault tolerance.
- Disaster recovery.

High volume services require efficiency of resource use. High availability requires, on the other hand, both efficiency and a reliability strategy that is fully integrated with the software deployment strategy. Typically one uses load sharing and redundancy for this.

5.2 Scalability

Like security, scalability is a property of systems that researchers and designers like to claim or even boast, backed up with at best spurious evidence, as if the mere mention of the word could bring fortune. Like security, it has become a word to mistrust. Part of the problem is that very little attention has been given to defining what scalability means. This chapter is not the place for a comprehensive discussion of scalability, but we can summarize some simple issues.

Scalability is about characterizing the output of a system as a function of input (usually a task). The ability of a system to complete tasks depends on the extent to which the task can be broken up into independent streams that can be completed in parallel, and thus the topology of the system and its communication channels. Some simple ideas about scalability can be understood using the flow approximation of queueing systems (see chapter of System Administration and the Scientific Method in this volume) to see how the output changes as we vary the input.

We shall think of scalability as the ability of a system to deal with large amounts of input, or more correctly, we consider how increasing the input affects the efficiency

with which tasks are discharged at the output. This sounds somewhat like the problem posed in queueing theory. However, whereas queueing theory deals with the subtleties of random processes, scalability is usually discussed in a pseudo-deterministic flow approximation.

If we think of rain falling (a random process of requests) then all we are interested in over time is how much rain falls and whether we can drain it away quickly enough by introducing a sufficient number of drains (processors or servers). Thus scalability is usually discussed as throughput as a function of load. Sometimes the input is a function of a number of clients, and sometimes the output is a function of the number of servers (see fig. 2). There are many ways to talk about scaling behaviour.

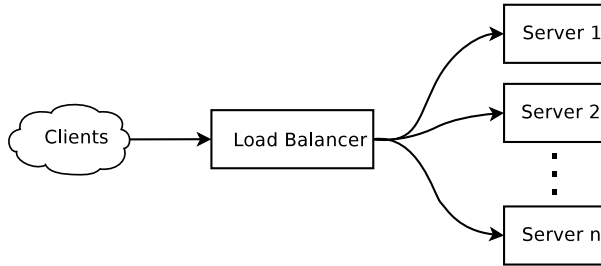


Figure 2: The topology for low level load sharing using a commercial dispatcher.

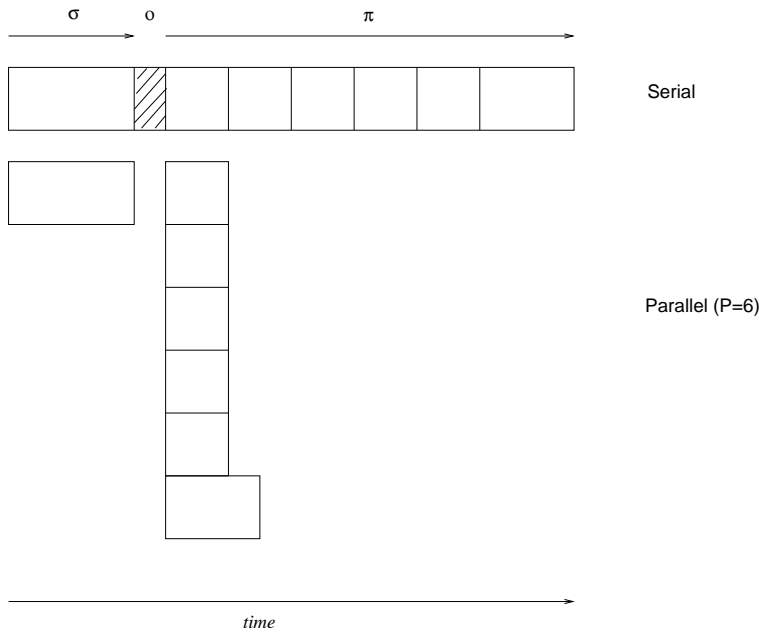


Figure 3: Representation of Amdahl's law. A typical task has only a fraction that is parallelizable. The serial part cannot be sped up using parallel processing or load balancing. Note that one is not always able to balance tasks into chunks of equal size, moreover there is an overhead (shaded) involved in the parallelization.

Amdahl's law, named after computer designer Gene Amdahl, was one of the first attempts to characterize scalability of tasks in High Performance Computing, in relation to the number of processors[1]. It calculates the expected “speed-up”, or fractional increase in performance, as a result of parallellizing part of the processing

(load balancing) between N processors as:

$$S = \frac{t_{\text{serial}}}{t_{\text{parallel}}} = \frac{t(1)}{t(N)} \quad (1)$$

Suppose a task of total size $\sigma + \pi$, which is a sum of a serial part σ and a parallelizable part π , is to be shared amongst N servers or processors and suppose that there is an overhead o associated with the parallelization (see fig. 3). Amdahl's law says that:

$$S = \frac{\sigma + \pi}{\sigma + \frac{\pi}{N} + o}. \quad (2)$$

In general, one does not know much about the overhead o except to say that it is positive ($o \geq 0$), thus we write

$$S \leq \frac{\sigma + \pi}{\sigma + \frac{\pi}{N}}. \quad (3)$$

This is usually rewritten by introducing units of the *serial fraction*, $f \equiv \sigma/(\sigma + \pi)$, so that we may write:

$$S \leq \frac{1}{f + \frac{(1-f)}{N}}. \quad (4)$$

This fraction is never greater than $1/f$, thus even with an infinite number of processors the task cannot be made faster than the processing of the serial part. The aim of any application designer must therefore be to make the serial fraction of an application as small as possible. This is called the bottleneck of the system.

Amdahl's law is, of course, an idealization that make a number of unrealistic assumptions, most notably that the overhead is zero. It is not clear that a given task can, in fact, be divided equally between a given number of processors. This assumes some perfect knowledge of a task with very fine granularity. Thus the speedup is strictly limited by the largest chunk (see fig. 3) not the smallest. The value of the model is that it predicts two issues that limit the performance of a server or high performance application:

- The serial fraction of the task³.
- The processor entropy or even-ness of the load balancing.

Amdahl's law was written with High Performance Computing in mind, but it applies also to server load balancing for network services. If one thinks of an entire job as the sum of all requests over a period of time (just as rain-water is a sum of all the individual rain-drops) then the law can also be applied to the speed up obtained in a load-balancing architecture such as that shown in fig. 2. The serial part of this task is then related to the processing of headers by the dispatcher, i.e. the dispatcher is the fundamental limitation of the system.

Determining the serial fraction of a task is not always as straightforward as one thinks. Contention for dependent resources and synchronization of tasks (waiting for resources to become available) often complicates the simple picture offered by the Amdahl law.

Karp and Flatt[21] noted that the serial part of a program must often be determined empirically. The Karp-Flatt metric is defined as the empirically determined

³It is often quoted that a single woman can have a baby in nine months, but nine women cannot make this happen in a month.

serial fraction, found by rearranging the Amdahl formula (4) to solve for the serial fraction.

$$f = \frac{S^{-1} - N^{-1}}{1 - 1/N}. \quad (5)$$

Here we assume no overhead (it will be absorbed into the final effective serial fraction). Note that, as the number of processors becomes large, this becomes simply the reciprocal of the measured speed up. This formula has no predictive power, but it can be used to measure the performance of applications with different numbers of processors. If S is small compared to N then we know that the overhead is large and we are looking for performance issues in the task scheduler.

Another way of looking at Amdahl's law in networks has been examined in refs. [7, 8] to discuss centralization versus distribution of processing. In network topology, serialization corresponds to centralization of processing, i.e. the introduction of a bottleneck by design. Sometimes this is necessary to collate data in a single location, other times designers centralize workflows unnecessarily from lack of imagination. If a centralized server is a common dependency of N clients, then it is clear that the capacity of the server C has to be shared between the N clients, so the workflow per client is

$$W \simeq \frac{C}{N}. \quad (6)$$

We say then that this architecture scales like $1/N$, assuming C to be constant. As N becomes large, the workflow per client goes to zero which is a poor scaling property. We would prefer that it were constant, which means that we must either scale the capacity C in step with N or look for a different architecture. There are two alternatives:

- Server scaling by load balancing (in-site or cross-site) $C \rightarrow CN$.
- Peer to peer architecture (client-based voluntary load balancing) $N \rightarrow 1$.

There is still much mistrust of non-centralized systems although the success of peer to peer systems is now hard to ignore. Scaling in the data centre cannot benefit from peer to peer scaling unless applications are designed with it in mind. It is a clear case where application design is crucial to the scaling.

5.3 Failure modes and redundancy

To gauge reliability system and software engineers should map out the failure modes of (or potential threats to) the system [6, 18] (see also the chapter on System Reliability in this volume). Basic failure modes include:

- Power supply.
- Component failure.
- Software failure (programming error).
- Resource exhaustion and thrashing.
- Topological bottlenecks.
- Human error.

So-called “single points of failure” in a system are warning signs of potential failure modes (see fig. 4). The principle of separation of concerns tends to lead to a tree-like structure which is all about *not* repeating functional elements and is therefore in basic conflict with the idea of redundancy (fig. 4a). To counter this, one can use dispatchers, load balancers and external parallelism (i.e. not built into the software, but implemented afterwards). This can cure some problems, but there might still be points of failure left over (fig. 4b). Ideally one tries to eliminate these through further redundancy, e.g. redundant internet service provision, redundant power supplies, etc. One should not forget the need for human redundancy in this reckoning: human resources and competence are also points of failure. Redundancy is the key issue in handling quality of service: it answers the issues of parallelism for efficiency and for fault tolerance.

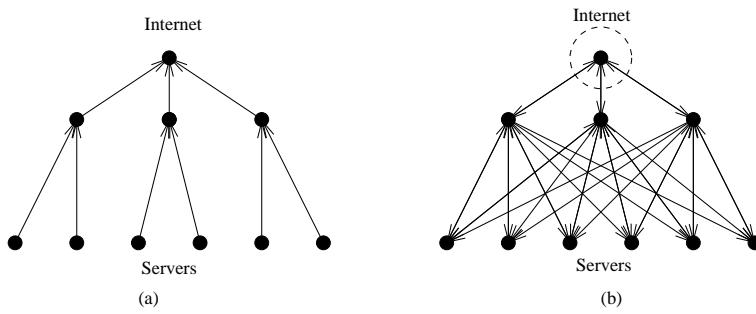


Figure 4: Load balancing is a tricky matter if one is looking for redundancy. A load balancer is a tree – which is a structure with many intrinsic points of failure and bottlenecks.

When planning redundancy, there are certain thumb rules and theorems concerning system reliability. Fig. 5 illustrates the folk-theorem about parallelisms which says that redundancy at lower system levels is always better than redundancy at higher levels. This follows from the fact that a single failure at a low level could

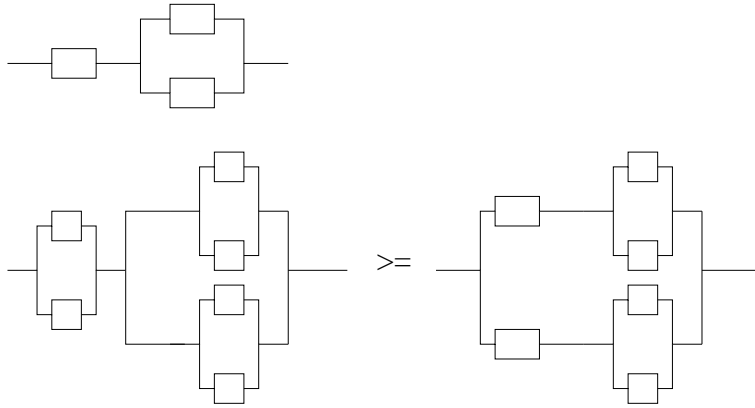


Figure 5: Redundancy folk theorem.

stop an entire computing component from working. With low-level redundancy, the weakest link only brings down a low level component, with high level redundancy, a weakest link could bring down an entire unit of dependent components.

A minimum dependency and full redundancy strategy is shown in fig. 6. Each of the doubled components can be scaled up to n to increase throughput.

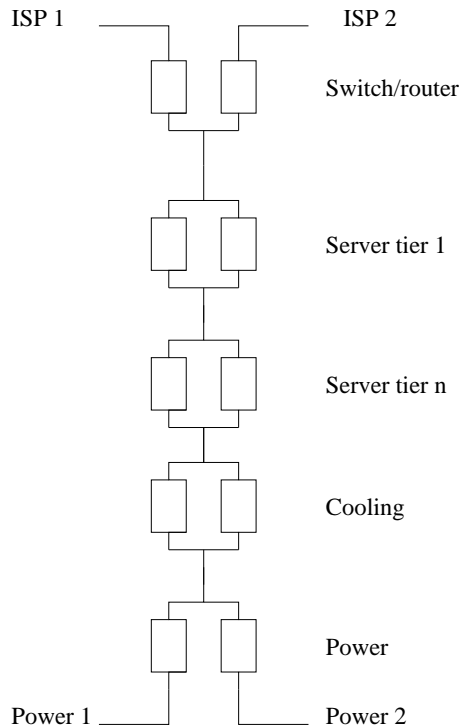


Figure 6: Minimum strategy for complete redundancy.

Security is another concern that can be addressed in terms of failure modes. It is often distinguished from system failure due to “natural causes” by being a “soft failure”, i.e. a failure relative to a policy rather than continuation. The distinction is probably unnecessary, as we can simply consider all failures relative to policy.

A secure system can be defined as follows[6]: *A secure system is one in which all the risks have been identified and accepted as a matter of policy.* See also the chapter by Bishop in this volume. This definition underlines the point that there is always an arbitrary threshold (or policy) in making decisions, about when faults are sufficiently serious to warrant a change of attitude, or a response.

An important enemy in system reliability is *human error*, either by mistake or incompetence. Human fault paths deal with many issues. Hostile parties expose us to risk through:

- Greed.
- Vanity.
- Bribery and blackmail.
- Revenge.
- Vandalism.
- Warfare.

Friends, on the other hand, expose us to risk by:

- Forgetfulness.
- Misunderstanding/miscommunication.

- Confusion/stress/intoxication.
- Arrogance.
- Ignorance / carelessness.
- Slowness of response.
- Procedural errors.
- Inability to deal with complexity.
- Inability to cooperate with others.

The lists are long. It is also worth asking why such incompetence could be allowed to surface. Systems themselves are clearly at fault in many ways through:

- Design faults – inadequate specification.
- Run-time fault – system does not perform within its specification.
- Emergent faults – behaviour that was not planned for or explicitly designed for, often provoked by the environment.
- Changing assumptions – about technology or the environment of the system, e.g. a system is designed for one scenario that becomes replaced by another.

Documentation of possible fault modes should be prioritized by their relative likelihood.

5.4 Safe and reliable systems

The construction of safe systems[5, 6] is both controversial and expensive. Security has many interpretations, e.g. see ISO17799[3] and RFC 2196 (replacing RFC1244), or The Orange Book Trusted Computer Security Evaluation Criteria (TSEC) (now somewhat dated).

Security, like fault analysis, begins with a threat evaluation. It includes issues from human interface design to resistance to hostile actions. In all cases a preventative response is a *risk reducing strategy*.

There are many unhealthy attitudes to security amongst system designers and even administrators, e.g. "Encryption solves 90% of the world's security problems" as stated by a naive software analyst known to the author. Security is a property of complete systems, not about installable features. Encryption deals with only a risk of information capture, which is one part of a large problem that is easily dealt with. Encryption key management, on the other hand, is almost never addressed seriously by organizations, but is clearly the basis for the reliability of encryption as a security mechanism.

Just as one must measure fault issues relative to a policy for severity, security requires us to place *trust boundaries*. The basis of everything in security is what we consider to be trustworthy (see the definition in the previous section). Technology can shift the focus and boundaries of our trust, but not eliminate the assumption of it. A system policy should make clear where the boundaries of trust lie. Some service customers insist on source code-review of applications as part of their Service Agreement. Threats include:

- Accidents.
- Human error, spilled coffee, etc.

- Attacks.
- Theft, spying, sabotage.
- Pathogenic threats like virus, Trojan horse, bugs.
- Human-computer interfaces.
- Software functionality.
- Algorithms.
- Hardware.
- Trust relationships (the line of defense).

Restriction of privilege is one defense against these matters. Access is placed on a "need to know/do" basis and helps one to limit the scope of damage. System modelling is an important tool in predicting possible failure modes (see chapters on System Administration and the Scientific Method by the author, and the chapter on Security Management and Policies by Bishop).

In short, a well-designed system fails in a predictable way, and allows swift recovery.

6 Data Centre Design

The data centre is a dry and noisy environment. Humans are not well suited to this environment and should try to organize work so that they do not spend more time there than necessary. Shoes and appropriate clothing shall be worn in data centres for personal safety and to avoid contaminating the environment with skin, hair and lost personal items. Un-authorized persons should never be admitted to the data centre for both safety and security reasons.

Data centres are not something that can be sold from a shelf; they are designed within the bounds of a specific site to cope with a specific tasks. Data centres are expensive to build, especially as performance requirements increase. One must consider power supply, cooling requirements, disasters (flooding, collision damage etc) and redundancy due to routine failure. This requires a design and requirement overview that companies rarely have in advance. The design process is therefore of crucial importance.

It is normal for inexperienced system administrators and engineers to underestimate the power and cooling requirements for a data centre. Today, with blade chassis computers and dense rack mounting solutions it is possible to pack even more computers into a small space than ever before, thus the density of heat and power far exceeds what would be required in a normal building. It is important to be able to deliver peak power during sudden bursts of activity without tripping a fuse or circuit-breaker.

With so much heat being generated and indeed wasted in such a small area, we have to think seriously in the future about how buildings are designed. How do we re-use the heat? How can cooling be achieved without power-driven heat-exchangers?

6.1 Power in the data centre

Power infrastructure is the first item on the agenda when building a data centre. Having sufficient access to electrical power (with circuit redundancy) is a prerequisite for stable operation. Data centres are rated in a tier system (see section 6.10) based on how much redundancy they can provide.

Every electrical device generates heat, including cooling equipment. Electrical power is needed to make computers run. It is also needed for lighting and cooling. Once power needs have been estimated for computers, we might have to add up to 70% again for the cost of cooling, depending on the building. In the future buildings could be designed to avoid this kind of gratuitous compounding of the heat/energy problem with artificial air-conditioning by cooling naturally with underground air intakes.

Power consumption is defined as the flow of energy, per unit time, into or out of a system. It is measured in Joules per second, or Watts or Volt-Amperes, all of which are equivalent in terms of engineering dimensions. However, these have qualitatively different interpretations for alternating current sources.

Computers use different amounts of electrical current and power depending on where they are located. Moreover, since electricity companies charge by the current that is drawn rather than the power used, the current characteristics of a device are important as a potential source of cost saving.

Electronic circuitry has two kinds of components:

- *Resistive components* absorb and dissipate power as heat. They obey Ohm's law.
- *Reactive components* absorb and return energy (like a battery), by storing it in electromagnetic fields. They include capacitors (condensers) and inductors (electro-magnetic coils).

Reactive components (inductors and capacitors) store and return most of the power they use. This means that if we wait for a full cycle of the alternating wave, we will get back most of what we put in again (however, we will still have to pay for the current). Resistive components, on the other hand, convert most of the energy going into them into heat.

In direct current (DC) circuits, the power (energy released per unit time) is simply given by $P = IV$ where I is the constant current and V is the constant voltage. In a device where a DC current can flow, reactive components do not have an effect and Ohm's law applies: $V = IR$, where R is the electrical resistance. In this case, we can write

$$P = IV = I^2 R = V^2 / R. \quad (7)$$

For an alternating current (wave) source, there are several cases to consider however. Ohm's law is no longer true in AC circuits, because capacitors and inductors can borrow power for a short time and then return it, like a wave shifting pebbles on a beach. The voltage and current are now functions of time: $I(t), V(t)$. The instantaneous power consumption (the total energy absorbed per unit time) is still $P = IV = I(t)V(t)$. However, mains power is typically varying at 50-60 Hz, so this is not an true reflection of the long term behaviour, only what happens on the scale of tenths of a second.

For a system driven by a single frequency (clean) wave, the short term borrowing of current is reflected by a phase shift between the voltage (wave) and the current, which is the response of the system to that driving force (pebbles). Let us call this phase shift ϕ .

$$\begin{aligned} V(t) &= V_0 \sin(2\pi ft) \\ I(t) &= I_0 \sin(2\pi ft + \phi) \end{aligned} \quad (8)$$

where f frequency and $T = 1/f$ is the period of the wave. We can compute the average power over a number of cycles nT by computing

$$\langle P \rangle_\phi = \frac{1}{nT} \int_0^{nT} I(t) V(t) dt \quad (9)$$

We evaluate this using two trigonometric identities:

$$\sin(A + B) = \sin A \cos B + \cos A \sin B \quad (10)$$

$$\sin^2 X = \frac{1}{2}(1 - \cos 2X) \quad (11)$$

Using the first of these to rewrite $I(t)$, we have

$$\langle P \rangle = \frac{I_0 V_0}{nT} \int_0^{nT} \left[\sin^2 \left(\frac{2\pi t}{T} \right) \cos \phi + \sin \left(\frac{2\pi t}{T} \right) \cos \left(\frac{2\pi t}{T} \right) \sin \phi \right] dt \quad (12)$$

Rewriting the first term with the help of the second identity allows us to integrate the expression. Most terms vanish showing the power that is returned over a whole cycle. We are left with:

$$\langle P \rangle_\phi = \frac{1}{2} I_0 V_0 \cos \phi. \quad (13)$$

In terms of the normally quoted root-mean-square (RMS) values:

$$V_{\text{rms}} = \sqrt{\frac{1}{nT} \int_0^{nT} (V(t))^2 dt} \quad (14)$$

For a single frequency one has simply:

$$V_{\text{rms}} = V_0 / \sqrt{2} \quad (15)$$

$$I_{\text{rms}} = I_0 / \sqrt{2}, \quad (16)$$

$$\langle P \rangle_\phi = I_{\text{rms}} V_{\text{rms}} \cos \phi.$$

The RMS values are the values returned by most measuring devices and they are the values quoted on power supplies. The cosine factor is sometimes called the “power factor” in electrical engineering literature. It is generalized below. Clearly

$$\langle P \rangle_{\phi > 0} \leq \langle P \rangle_{\phi = 0}. \quad (17)$$

In other words, the actual power consumption is not necessarily as bad as the values one would measure with volt and ammeters. This is a pity when we pay our bill, because the power companies measure current, not work-done. That means we typically pay more than we should for electrical power. Large factories can sometimes negotiate discounts based on the reactive power factor.

6.2 Clipped and dirty power

A direct calculation of the reduction in power transfer is impractical in most cases, and one simply defines a power factor by

$$PF = \cos \phi \equiv \frac{\langle P \rangle}{I_{\text{rms}} V_{\text{rms}}} \quad (18)$$

More expensive power supply equipment is designed with ‘power factor corrected’ electronics that attempt to reduce the reactance of the load and lead to a simple $\phi = 0$ behaviour. The more equipment we put onto a power circuit, the more erratic the power factor is likely to be. This is one reason to have specialized power supplies (incorporated with Uninterruptible Power Supplies (UPS)).

Actual data concerning power in the computer centres is hard to find, so the following is based on hearsay. Some authors claim that power factors of $\cos \phi = 0.6$

have been observed in some PC hardware. Some authors suggest that a mean value of $\cos \phi = 0.8$ is a reasonable guess. Others claim that in data centres one should assume that $\cos \phi = 1$ to correctly allow for enough headroom to deal with power demand.

Another side effect of multiple AC harmonics is that the RMS value of current is not simply $1/\sqrt{2} = 1/1.4$ of the peak value (amplitude). This leads to a new ratio called the “crest factor”, which is simply:

$$\text{Crest} = \frac{I_0}{I_{\text{rms}}}. \quad (19)$$

This tells us about current peaking during high load. For a clean sinusoidal wave, this ratio is simply $\sqrt{2} = 1.4$. Some authors claim that the crest factor can be as high as 2-3 for cheap computing equipment. Some authors claim that a value of 1.4-1.9 is appropriate for power delivered by a UPS (uninterruptible power supply). The crest factor is also load dependent.

6.3 Generators and batteries

Uninterruptible Power Supplies (UPS) serve two functions: first to clean up the electrical power factor, and second to smooth out irregularities including power outages. Batteries can take over almost instantaneously from supply current, and automatic circuit breakers can start a generator to take over the main load within seconds. If the generator fails, then battery capacity will be drained.

UPS devices require huge amounts of battery capacity and even the smallest of these weighs more than a single person can normally lift. They must be installed by a competent electrician who knows the electrical installation details for the building.

Diesel powered generators should not be left unused for long periods of time, as bacteria, molds and fungi can live in the diesel and transform it into jelly over time.

6.4 Cooling and Airflow

Environmental conditions are the second design priority in the data centre. This includes cooling and humidity regulation. Modern electronics work by dumping current to ground through semi-conducting (resistive) materials. This is the basic transistor mode of operation. This generates large amounts of heat. Essentially all of the electrical power consumed by a device ultimately becomes heat.

The aim of cooling is to prevent the temperature of devices becoming too great, or changing too quickly. If the heat increases, the electrical and mechanical resistance of all devices increases and even more energy is wasted as heat. Eventually, the electrical properties of the materials will become unsuitable for their original purpose and the device will stop working. In the worst case it could even melt.

Strictly speaking, heat is not dangerous to equipment, but temperature is. Temperature is related to the density of work done. The more concentrated heat is, the higher the temperature. Thus we want to spread heat out as much as possible.

In addition to the peak temperature, changes in temperature can be dangerous in the data centre. If temperature rises or falls too quickly it can result in mechanical stress (expansion and contraction of metallic components), or condensation in cases of high humidity.

- Polymer casings and solder-points can melt if they become too hot.
- Sudden temperature changes can lead to mechanical stresses on circuits, resulting in cracks in components and circuit failures.

- Heat increases electrical and mechanical resistance, which in turn increases heat production since heat power conversion goes like $\sim I^2 R$.

Air, water and liquefied gases can be used to cool computing equipment. Which of these is best depends on budget and local environment. Good environmental design is about the constancy of the environment. Temperature, humidity, power consumption and all variables should be evenly regulated in space and time.

6.5 Heat design

Cooling is perhaps the most difficult aspect of data centre design to get right. The flow of air is a complicated science and our intuitions are often incorrect. It is not necessarily true that increasing the amount of cooling in the data centre will lead to better cooling of servers.

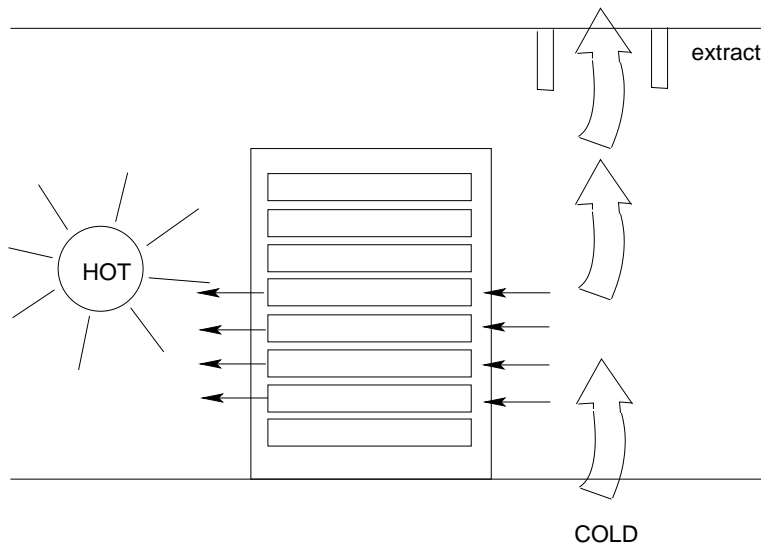


Figure 7: A bad cooling design. Too much cold air comes through the floor, driving all the cold air past the racks into the extractor instead of passing through them. This leaves hot spots and wastes most of the cooling capacity.

Cooling equipment is usually rated in BTUs (British Thermal Units), an old fashioned measurement of heat. 1 Watt = 3413 BTUs. A design must provide cooling for every heat generating piece of equipment in the data centre, including the humans and the cooling equipment itself. Temperature must be regulated over both space and time, to avoid gradient effects like condensation, disruptive turbulence, and heat-expansion or contraction of components.

The key to temperature control is to achieve a constant ambient temperature throughout a room. The larger a room is, the harder this problem becomes. If hot spots or cold spots develop, these can lead to problems of condensation of moisture, which increases in likelihood with increasing humidity and is both harmful to the computing equipment and can even lead to bacterial growth which is a health hazard (e.g. Legionnaires disease).

If air moves too quickly past hot devices, it will not have time to warm up and carry away heat from hot spots; in this case, most of the air volume will simply cycle from and back to the cooling units without carrying away the heat. If air moves too slowly, the temperature will rise.

As everyone knows, hot air rises if it is able to do so. This can be both a blessing and a curse. Raised flooring is a standard design feature in more advanced data

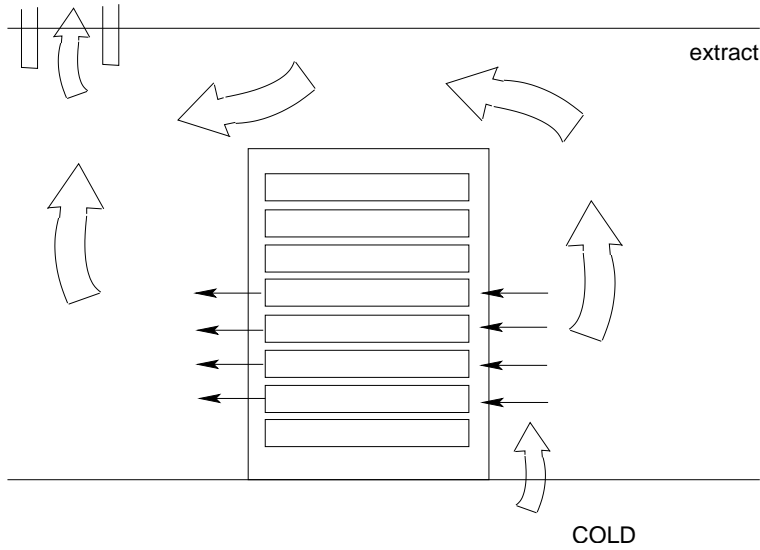


Figure 8: A good cooling design. A trickle cold air comes through the floor, spreading out near the racks into the room where has time to collect heat. The air circulates from the cold aisle into the warm aisle where it is extracted and recycled.

centres which allows cool air to come up through through holes in the floor. This is a useful way of keeping air moving in the data centre so that no hot spots can develop. However, one should be careful not to have cold air entering too quickly from below, as this will simply cause the hot air to rise into the roof of the centre where it will be trapped. This will make some servers too hot and some too cold. It can also cause temperature sensors to be fooled into misreading the ambient temperature of the data centre, causing energy wastage from over cooling, or over regulation of cooling.

Over-cooling can, again, lead to humidity and condensation problems. Large temperature gradients can cause droplets of water to form (as in cloud formation). Hot computer equipment should not be placed too close to cooling panels as this will cause the coolers to work overtime in order to cool the apparent imbalance, and condensation-ice can form on the compressors. Eventually they will give up and water can then flood into the data centre under the flooring. The solution, unfortunately, is not merely to dry out the air, as this can lead to danger of static electrical discharges and a host of related consequences.

6.6 Heat, temperature and humidity

Heat and temperature are related through the specific heat capacity (symbol c_s , also called specific heat) of a substance is defined as heat capacity per unit mass. The SI unit for specific heat capacity is the Joule per kilogramme Kelvin, $\text{Jkg}^{-1}\text{K}^{-1}$ or $\text{J}/(\text{kgK})$, which is the amount of energy required to raise the temperature of one kilogram of the substance by one Kelvin. Heat capacity can be measured by using calorimetry.

For small temperature variations one can think of heat capacity as being a constant property of a substance, and a linear formula for the relates temperature and heat energy.

$$\Delta Q = mc_s \Delta T \quad (20)$$

where ΔQ is a change of heat energy (Watts \times time). In other words temperature

increase ΔT , in a fixed mass m of any substance, is proportional to the total power output and the time devices are left running.

If air humidity is too high, it can lead to condensation on relatively cool surfaces. Although a small amount of clean, distilled water is not a serious danger to electrical equipment, water mixed with dust, airborne dirt or salts will conduct electricity and can be hazardous.

Computer equipment can work at quite high temperatures. In ref. [24], experienced engineers suggest that the maximum data centre temperature should be 25 degrees Celcius (77 Fahrenheit), with a relative humidity of at least 40% (but no more than 80%). At less than 30% humidity, there is a risk of static electrical discharge, causing read errors or even unrecoverable runtime failures.

Heat is transmitted to and from a system by three mechanisms:

- *Conduction* is the direct transfer of heat by material contact (molecular vibration). For this, we use the heat capacity formula above.
- *Convection* is a transport of gas around a room due to the fact that, as a gas is heated its density decreases under constant pressure, so it gets lighter and rises. If the hot air rises, cooler air must fall underneath it. This results in a circulation of air called convection. If there is not enough space to form convection cells, hot air will simply sit on top of cold air and build up.
- *Radiation* is the direct transfer of heat without material contact. Electromagnetic radiation only passes through transparent substances. However, a substance that is transparent to visible light is not necessarily transparent to heat (infra-red), which has a much longer wavelength. This is the essence of the Greenhouse effect. Visible light can enter a Greenhouse through the glass walls, this energy is absorbed and some of the energy is radiated back as heat (some of it makes plants grow). However not all of the long wavelength heat energy passes through the glass. Some is absorbed by the air and heats it up. Eventually, it will cool off by radiation of very long wavelengths, but this process is much slower than the rate at which new energy is coming in, hence the energy density increases and temperature goes up.

If a material is brought into contact with a hotter substance, heat will flow into it by *conduction*. Then, if we can remove the hotter substance e.g. by flow convection, it will carry the heat away with it. Some values of c_s for common cooling materials are:

Substance	Phase	Specific heat capacity
Air (dry)	gas	1005 J/kg/K
Air (100% humidity)	gas	1030 J/kg/K
Air (different source)	gas	1158 J/kg/K
Water	liquid	4186 J/kg/K
Copper (room temp)	solid	385 J/kg/K
Gold (room temp)	solid	129 J/kg/K
Silver (room temp)	solid	235 J/kg/K

Water is about four times as efficient at carrying heat per degree rise in temperature than is air. It is therefore correspondingly better at cooling than air.

Melting temperatures below show that it is not likely that data centre heat will cause anything in a computer to melt. The first parts to melt would be the plastics and polymers and solder connections:

Material	Melting Point
Silicon	1410 C
Copper	1083 C
Gold	1063 C
Silver	879 C
Polymers	50-100 C

Today's motherboards come with temperature monitoring software and hardware which shuts the computer off the CPU temperature gets too hot. CPU maximum operating temperatures lie between 60 and 90 degrees Celcius.

6.7 Cooling fans

Fans are place in computers to drive air through them. This air movement can also help to keep air circulating in the data centre. Rack doors and poorly placed fans can hinder this circulation by generating turbulence.

Cooling fans exploit the same idea as convection, to transport energy away by the actual movement of air (or sometimes fluid). The rate of flow of volume is

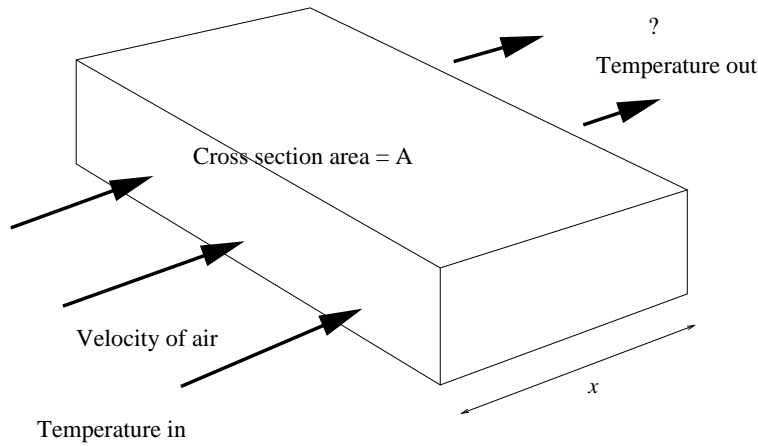


Figure 9: Schematic illustration of air flowing through and around a rack device.

proportional to the fixed area of the channel cross-section and the length per unit time (velocity) of the coolant passing through it.

$$\frac{dV}{dt} \propto A \frac{dx}{dt} \quad (21)$$

$$F \propto Av \quad (22)$$

You should bear in mind that

- It is the temperature of components inside the computer that is the main worry in cooling design.
- The temperature of the air coming out depends on many factors, including the ambient temperature around the box, the rate of air flow through the device and around it, the amount of convection in and around the device. humidity etc.
- Air is a poor carrier of heat, so if air is getting hot, you know that the metallic components are sizzling! Also, air is a good insulator so it takes some time to warm up. If air moves too fast, it will not carry away the heat.

In very high speed computers that generate large amounts of heat quickly, liquids such as water and nitrogen are used for cooling, piped in special heat exchangers. The density of a liquid means that it is able to carry much more heat, and hence a smaller flow is needed to carry a larger amount of waste power. Water-based cooling is many times more expensive than air-cooling.

6.8 Placement of cooling flow ducts

The use of raised flooring is common in datacentres. Cold air usually passes underneath raised flooring from compressors and escapes into the data centre through holes in the floor.

If cold air rose from everywhere quickly and evenly, hot air would simply rise to the roof and there would be no circulation (see fig. 7). A standard strategy to maintain convectional circulation is to limit the number of holes in flooring to prevent air from circulating too quickly, and then have alternating hot and cold aisles. In a cold aisle, there are holes in the floor. In a hot aisle there are no holes in the floor, but there are cooling intakes (see fig. 8).

To ensure sufficient circulation, there should be at least ten centimetres of clearance between racks in the roof and computer racks should not be within a metre or more of cooling units.

6.9 Fire suppression

A data center fire suppression system uses gas rather than water or foam. There are many gaseous suppression systems that use gases hazardous to humans and environment and are therefore unsuitable. The gas used to suppress the fire should not be toxic to people nor damage the equipment causing data to be lost. Inergen and Argonite are both gasses commonly used in data centers. Inergen is a gaseous mixture composed of nitrogen, argon and carbon dioxide. Argonite is a gas composed of argon and nitrogen. Since the fire suppression system is based on gas one should make sure that in case of gas release there is no leakage to the outside.

6.10 Tier Performance Standards

The Uptime Institute has created a 4 Tier rating system of a data centre[33]. Tier 1 level is the most basic where no redundancy is required, while Tier 4 level is a 100% redundant data center. A system is as good as the weakest link and hence terms like “near 4 Tier or “Tier 3 plus do not exist.

The ratings use the phrases *capacity components* and *distribution paths*. Capacity components are active components like servers which deliver the service provided by the data centre. Distribution paths refer to communication infrastructure, such as network.

The Tier 1 describes a basic site, i.e. a non-redundant data center. Communication paths, cabling, power and cooling are non-redundant, the computer system is affected by a component failure and the system has to be shut down to do maintenance work.

Tier 2 describes a data center that has redundant capacity components, but non-redundant distribution paths. The computer system might be affected by a capacity component failure, and will certainly be affected by a path failure. The system might be inoperative during maintenance work.

Tier 3 describes a concurrently maintainable data center. In other words, the data center is equipped with redundant capacity components and multiple distribution paths (of which one is always active). The computer system will not be

affected either in case of a capacity component failure or path failure and does not require shutdown to do maintenance work, but interruptions could occur.

Tier 4 describes a fault tolerant data center. In other words, the data center is equipped with redundant capacity components and multiple distribution active paths. The computer system will not be affected of any worst-case failure of any capacity system or distribution element and does not require to be shut down to do maintenance work, no interruptions may occur.

7 Capacity Planning

Simple results from queueing theory can be used as estimators of the capacity needs for a system [2, 9]. Most service delivery systems are I/O bound, i.e. the chief bottleneck is input/output. However, in a data centre environment we have another problem that we do not experience on a small scale: power limitations.

Before designing a data center we need to work out some basic numbers, the scales and sizes of numbers involved.

- How fast can we process jobs? (CPU rate and number of parallel servers).
- How much memory do we need: how many jobs will be in the system (queue length) and what is the average size of a job in memory?
- How many bytes per second of network arrivals and returns do we expect?
- How many bytes per second of disk activity do we expect?
- What is the power cost (including cooling) of running out servers?

7.1 Basic queueing theory

Service provision is usually modelled as a relationship between a client and a server (or service provider). A stream of requests arrives randomly at the input of a system, at a mean rate of λ transactions per second, and they are added to a queue, whereupon they are serviced at a mean rate μ transactions per second by a processor.

Generalizations of this model include the possibility of multiple (parallel) queues and multiple servers. In each case one considers a single incoming stream of transaction requests; one then studies how to cope with this stream of work.

As transaction demand is a random process, queues are classified according to the type of arrival process for requests, the type of completion process and the number of servers. In the simplest form, Kendall notation of the form $A/S/n$ is used to classify different queueing models.

- A : the arrival process distribution, e.g. Poisson arrival times, deterministic arrivals, or general spectrum arrivals.
- S : the service completion distribution for job processing, e.g. Poisson renewals, etc.
- n : the number of servers processing the incoming queue.

The basic ideas about queueing can be surmised from the simplest model of a single server, memoryless queue: $M/M/1$ (see chapter by Burgess, System Administration and the Scientific Method in this volume).

Queues are described in terms of the statistical pattern of job arrivals. These processes are often approximated by so-called memoryless arrival processes with

mean arrival rate λ jobs per second and mean processing rate μ jobs per second. The quantity $\rho = \lambda/\mu < 1$ is called the traffic intensity[6, 19]. The expected length of the simplest M/M/1 queue is

$$\langle n \rangle = \sum_{n=0}^{\infty} p_n n = \frac{\rho}{1-\rho}. \quad (23)$$

Clearly as the traffic intensity ρ approaches unity, or $\lambda \rightarrow \mu$, the queue length grows out of control, as the server loses the ability to cope.

The job handling improves somewhat for s servers (here represented by the M/M/s queue), where one finds a much more complicated expression which can be represented by:

$$\langle n \rangle = s\rho + P(n \geq s) \frac{\rho}{1-\rho} \quad (24)$$

The probability that the queue length exceeds the number of servers $P(n \geq s)$ is of order ρ^s for small load ρ , which naturally leads to smaller queues.

It is possible to show that a single queue with s servers is always at least as efficient as s separate queues with their own server. This satisfies the intuition that a single queue can be kept moving by any spare capacity in any of its s servers, whereas an empty queue that is separated from the rest will simply be wasted capacity, while the others struggle with the load.

7.2 Flow laws in the continuum approximation

As we noted, talking about scalability, in the memoryless approximation we can think of job queues as fluid flows. Dimensional analysis provides us with some simple continuum relationships for the ‘workflows’. Consider the following definitions:

$$\text{Arrival rate } \lambda = \frac{\text{No. of arrivals}}{\text{Time}} = \frac{A}{T} \quad (25)$$

$$\text{Throughput } \mu = \frac{\text{No. of completions}}{\text{Time}} = \frac{C}{T} \quad (26)$$

$$\text{Utilization } U = \frac{\text{Busy time}}{\text{Total time}} = \frac{B}{T} \quad (27)$$

$$\text{Mean service time } S = \frac{\text{Busy Time}}{\text{No. of completions}} = \frac{B}{C}. \quad (28)$$

The utilization U tells us the mean level at which resources are being scheduled in the system. The ‘utilization law’ notes simply that:

$$U = \frac{B}{T} = \frac{C}{T} \times \frac{B}{C} \quad (29)$$

or

$$U = \mu S. \quad (30)$$

So utilization is proportional to the rate at which jobs are completed and the mean time to complete a job. Thus it can be interpreted as the probability that there is at least one job in the system. Often this law is written $U = \lambda S$, on the assumption that in a constant flow the flow rate in is equal to the flow rate out of the system $C/T \rightarrow \lambda$. This is reasonable at low utilization because μ is sufficiently greater than λ that the process can be thought of as deterministic, thus:

$$U = \lambda S. \quad (31)$$

This handwaving rule is only applicable when the queue is lightly loaded and all jobs are essentially completed without delay; e.g. suppose a web server receives hits at a mean rate of 1.25 hits per second, and the server takes an average of 2 milliseconds to reply. The law tells us that the utilization of the server is

$$U = 1.25 \times 0.002 = 0.0025 = 0.25\%. \quad (32)$$

This indicates to us that the system could probably work at four hundred times this rate before saturation occurs, since $400 \times 0.25 = 100\%$. This conclusion is highly simplistic, but gives a rough impression of resource consumption.

Another dimensional truism is known as Little's law of queue size. It says that the mean number of jobs in a queue $\langle n \rangle$, is equal to the product of the mean arrival rate λ (jobs per second) and the mean response time R (seconds) incurred by the queue:

$$\langle n \rangle = \lambda R. \quad (33)$$

Note that this equation has the generic form $V = IR$, similar to Ohm's law in electricity. This analogy is a direct consequence of a simple balanced flow model. Note that R differs from the mean service time. R includes the waiting time in the queue, whereas the mean service time assumes that the job is ready to be processed.

In the $M/M/1$ queue, it is useful to characterize the expected response time of the service centre. In other words, what is the likely time a client will have to wait in order to have a task completed? From Little's law, we know that the average number of tasks in a queue is the product of the average response time and the average arrival rate, so

$$\begin{aligned} R = \frac{Q_n}{\lambda} &= \frac{\langle n \rangle}{\lambda} \\ &= \frac{1}{\mu(1 - \rho)} \\ &= \frac{1}{(\mu - \lambda)}. \end{aligned} \quad (34)$$

Notice that this is finite as long as $\lambda \ll \mu$, but as $\lambda \rightarrow \mu$, the response time becomes unbounded.

Load balancing over queues is a topic for more advanced queueing theory. Weighted fair queueing and algorithms like round-robin etc., can be used to spread the load of an incoming queue amongst multiple servers, to best exploit their availability. Readers are referred to refs. [6, 19, 13, 35] for more discussion on this.

7.3 Simple queue estimates

The $M/M/1$ queue is simplistic, but useful for its simplicity. Moreover, there is a point of view amongst system engineers that says: most systems will have a bottleneck (a point at which there is a single server queue) somewhere, and so we should model weakest links as $M/M/1$. Let's use the basic $M/M/1$ formulae to make some order-of-magnitude calculations for capacity planning.

For example, using the formulae for the $M/M/1$ queue, let's calculate the amount of RAM, disk rate and CPU speed to handle the average loads:

7.3.1 Problem

1. 10 downloads per second of image files 20MB each.
2. 100 downloads per second of image files 20MB each.

3. 20 downloads per second of resized images to a mobile phone, each image is 20MB and takes 100 CPU cycles per byte to render and convert in RAM.

given that the size of the download request is of the order 100 bytes.

7.3.2 Estimates

1. If we have 10 downloads per second, each of 20 MB/s, then we need to process

$$10 \times 20 = 200MB/s. \quad (35)$$

Ideally, all of the components in the system will be able to handle data at this rate or faster. (Note that when the averages rates are equal ($\lambda = \mu$), the queue length is not zero but infinite as this means that there will be a significant probability that the stochastic arrival rate will be greater than the stochastic processing rate). Let us suppose that the system disk has an average performance of 500 MB/s, then we have:

$$\begin{aligned} \lambda &= \frac{200}{20} = 10j/s \\ \mu &= \frac{500}{20} = 25j/s. \end{aligned} \quad (36)$$

Note that we cannot use MB/s for λ , or the formulae will be wrong. In the worst case, each component of the system must be able to handle this data rate. The average queue length is:

$$\langle n \rangle = \frac{\rho}{1 - \rho} = \frac{\lambda}{\mu - \lambda} = 2/3 \simeq 0.7 \quad (37)$$

This tells us that there will be about 1 job in the system queue waiting to be processed at any given moment. That means the RAM we need to service the queue will be 1×100 bytes, since each job request was 100 bytes long. Clearly RAM is not much of an issue for a lightly loaded server.

The mean response time for the service is

$$R = \frac{1}{\mu - \lambda} = \frac{1}{25 - 10} = \frac{1}{15} sec. \quad (38)$$

2. With the numbers above the server would fail long before this limit, since at queue size blows up to infinity as $\lambda \rightarrow 25$ and the server starts thrashing, i.e. it spends all its time managing the queue and not completing jobs.
3. Now we have an added processing time. Suppose we assume that the CPU has a clock speed of 10^9 Hz, and an average of 4 cycles per instruction. Then we process at a rate of 100 CPU cycles per byte, for all 20MB.

$$\frac{20 \times 10^6 \times 100}{10^9} = 2s. \quad (39)$$

Thus we have a processing overhead of 2 seconds per download, which must be added to the normal queue response time.

$$\begin{aligned} \lambda &= \frac{400}{20} = 20j/s \\ \mu &= \frac{500}{20} = 25j/s. \\ \langle n \rangle &= \frac{\rho}{1 - \rho} = \frac{\lambda}{\mu - \lambda} = 4 \\ R &= \frac{1}{\mu - \lambda} + CPU = \frac{1}{25 - 20} + CPU = \frac{1}{5} + 2sec. \end{aligned} \quad (40)$$

Notice that the CPU processing now dominates the service time, so we should look at increasing the speed of the CPU before worrying about anything else.

Developing simple order of magnitude estimates of this type forces system engineers to confront resource planning in a way that they would not normally do.

7.4 Peak load handling – the reason for monitoring

Planning service delivery based on average load levels is a common mistake of inexperienced service providers. There is always a finite risk that a level of demand will occur that is more than the system can provide for. Clearly we would like to avoid such a time, but this would require expensive use of redundant capacity margins (so called “over-provisioning”).

A service centre must be prepared to either provide a constant margin of headroom, or be prepared to deploy extra server power on demand. Virtualization is one strategy for the latter, as this can allow new systems to be brought quickly on line using spurious additional hardware. But such extended service capacity has to be planned for. This is impossible if one does not understand the basic behaviour of the system in advance (see the chapter on System Administration and the Scientific Method).

Monitoring is essential for understanding normal demand and performance levels in a system. Some administrators think that the purpose of monitoring is to receive alarms when faults occur, or for reviewing what happened after an incident. This is a dangerous point of view. First of all, by the time an incident has occurred, it is too late to prevent it. The point of monitoring ought to be to predict possible future occurrences. This requires a study of both average and deviant (“outlier”) behaviour over many weeks (see the chapter on System Administration and the Scientific Method).

The steps in planning for reliable provision include:

- Equipping the system for measurement instrumentation.
- Collecting data constantly.
- Characterizing and understand workload patterns over the short and the long term.
- Modelling and predicting performance including the likelihood and magnitude of peak activity.

Past data can indicate patterns of regular behaviour and even illuminate trends of change, but they cannot truly predict the future.

In a classic failure of foresight in their first year of internet service, the Norwegian tax directorate failed to foresee that there would be peak activity on the evening of the deadline for delivering tax returns. The system crashed and extra time had to be allocated to cope with the down-time. Such mistakes can be highly embarrassing (the tax authorities are perhaps the only institution that can guarantee that they won't lose money on this kind of outage).

Common mistakes of prediction include:

- Measuring only average workload without outlier stress points.
- Not instrumenting enough of the system to find the source of I/O bottlenecks.
- Not measuring uncertainties and data variability.
- Ignoring measurement overhead from the monitoring itself.

- Not repeating or validating measurements many times to eliminate random uncertainties.
- Not ensuring same conditions when making measurements.
- Not measuring performance under transient behaviour (rapid bursts).
- Collecting data but not analyzing properly.

8 Service Level Estimators

Service Level Agreements are increasingly important as the paradigm of service provision takes over the industry. These are legal documents, transforming offers of service into obligations. Promises that are made in service agreements should thus be based on a realistic view of service delivery; for that we need to relate engineering theory and practice.

8.1 Network Capacity Estimation

The data centre engineer would often like to know what supply-margin (“over-provision”) of service is required to be within a predictable margin of an Service Level Agreement (SLA) target? For example, when can we say that we are 95% certain of being able to deliver an SLA target level 80% of the time? Or better still: what is the full probability distribution for service within a given time threshold[2]?

Such estimates have to be based either on empirical data that are difficult to obtain, or simplified models that describe traffic statistics. In the late 1980’s and early 90’s it became clear that the classical Poisson arrivals assumptions which worked so well for telephone traffic were inadequate to describe Internet behaviour. Leland et al. [22] showed that Ethernet traffic exhibited self similar characteristics, later followed up in ref. [29]. Paxson et al. [28] found that application layer protocols like TELNET and FTP were modelled quite well by a Poisson model. However, the nature of data transfer proved to be bursty with long-range dependent behaviour.

Web traffic is particularly interesting. Web documents can contain a variety of in-line objects such as images and multimedia. They can consist of frames and client side script. Different versions of HTTP (i.e. version 1.0 and 1.1) coexist and interact. Implementations of the TCP/IP stack might behave slightly different, etc. depending on the operating system. This leads to great variability at the packet level[12]. Refs. [14, 27] have suggested several reasons for the traffic characteristics.

In spite of the evidence that for self-similarity of network traffic, it is known that if one only waits for long enough, one should see Poisson behaviour[20]. This is substantiated by research done in [10] and elsewhere, and is assumed true in measurements approaching the order of 10^{12} points. However this amounts to many years of traffic[17].

8.2 Server performance modelling

To achieve a desired level of quality of service under any traffic conditions, extensive knowledge about how web server hardware and software interacts and affects each other is required. Knowing the expected nature of traffic and queueing system only tells us how we might expect a system to perform; knowledge about software and hardware interaction enables performance tuning.

Slothouber [32] derived a simple model for considering web server systems founded on the notion of serial queues. Several components interact during a web conversation and a goes through different stages, with different queues at each stage. The

response time is therefore an aggregate of the times spent at different levels within the web server.

Mei et al. [34] followed this line of reasoning in a subsequent paper. One of the focal points in their work was to investigate the effects of for response time and server blocking probability due to congestion in networks. High end web servers usually sit on a network with excess bandwidth to be able to shuffle load at peak rates. However, customers inherently has poorly performing networks with with possible asynchronous transfer mode, such as ADSL lines. Therefore, returning ACKs in from client to server can be severely delayed. In turn this causes the request to linger for a longer time in the system than would be the case if the connecting network was of high quality. Aggregation of such requests could eventually cause the TCP and HTTP listen queue to fill up, making the server refuse new connection requests even if it is subject to fairly light load.

These models are simple and somewhat unrealistic, but possess the kind of simplicity we are looking for for our comparisons.

8.3 Service Level Distributions

The fact that both arrival and service processes are stochastic processes without a well-defined mean, combined with the effect of processor sharing has on response time tails, leads to the conclusion that there is an intrinsic problem in defining average levels for service delivery. Instead, it is more fruitful to consider the likelihood of being within a certain target. For instance, one could consider 80% or 90% compliance of a Service Level Objective. Depending on the nature of traffic the differences between these two levels can be quite significant.

This becomes quite apparent if we plot the Cumulative Distribution Functions (CDF) for the experimental and simulated response time distributions (see fig. 10)[2]. Because of the large deviations of values, going from 80% to 90% is not trivial from a service providers point of view. The CDF plots show that going from 80% to 90% in Service Level confidence means, more or less, a double of the service level target.

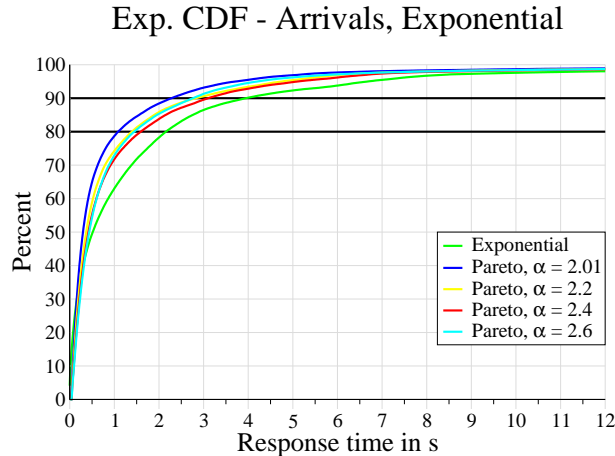


Figure 10: The cumulative percentage distribution for response processing for sample exponential and Pareto arrivals. The thick cut lines show the 80% and 90% levels, and the corresponding response times associated with them. The theoretical estimators are too pessimistic here, which is good in the sense that it would indicate a recommended over-capacity, but the value is surely too much.

The proper way to quote service levels is thus distributions. Mean response times and rates are useless, even on the time scale of hours. A cumulative probability plot of service times is straightforwardly derived from the experimental results and gives the most realistic appraisal of the system performance. The thick cut lines in the figure show the 80% and 90% levels, and the corresponding response times associated with them. From this one can make a Service Agreement based on probably compliance with a service time, i.e. not merely expectation value but histogram.

For instance, one could consider 80% or 90% compliance of a maximum service level target. Depending on the nature of traffic, the differences between these two levels can be quite significant. Since customers are interested in targets, this could be the best strategy available to the provider.

8.4 Over-capacity planning

The basic results for $M/M/1$ queues can be used to make over-estimates for capacity planning. The best predictive prescription (with noted flaws) is provided in ref. [2]:

1. Calculate the server rate μ for the weakest server that might handle a request.
e.g. for a CPU bound process

$$\begin{aligned}\mu &= \text{av. job size} \times \frac{\text{av. instructions}}{\text{job size}} \\ &\times \frac{\text{CPU cycles}}{\text{instruction}} \times \frac{\text{time}}{\text{cycles}} \\ &= \frac{\text{av.instructions} \times \text{RISC/CISC ratio}}{MHz}\end{aligned}\tag{41}$$

2. Estimate traffic type at maximum load.
3. Construct the Cumulative Probability Distribution as a function of response times using FCFS.
4. Make SLA promises about probability of transaction rate based on where the curves cross basic thresholds.

With this recipe, one overestimates service promise times to achieve a given level of certainty (or equivalently overestimates required capacity for a given time) in a non-linear way — the amount can be anything up to four hundred percent! This explains perhaps why most sites have seemingly greatly over-dimensioned web-servers. Given the problems that can occur if a queue begins to grow, and the low cost of CPU power, this over-capacity might be worth the apparent excess.

Readers are also referred to the cost calculations described in the chapter System Administration and the Business Process by Burgess, in this volume.

9 Network load sharing

Load sharing in the data centre is an essential strategy for meeting service levels in high volume and high availability services. Fig. 2 shows the schematic arrangement of servers in a load balancing scenario. Classical queueing models can also be used to predict the scaling behaviour of server capacity in a data centre.

Since service provision deals with random processes, exact prediction is not an option. A realistic picture is to end up with a probability or confidence measure e.g. what is the likelihood of being able to be within 80% or 90% of an SLA target value?

The shape of this probability distribution will also answer the question: what is the correct capacity margin for a service in order to meet the SLA requirements.

Once again, in spite of the limitations of simple queueing models, we can use these as simple estimators for service capacity[9]. The total quality of service in a system must be viewed as the combined qualities of the component parts[30]. It is a known result of reliability theory[18] that low level parallelism is, in general, more efficient than high level parallelism, in a component-based system. Thus a single $M/M/n$ queue is generally superior in performance to n separate $M/M/1$ queues (denoted $(M/M/1)^n$).

To understand load, we first have to see its effect on a single machine (fig 11). We see that a single computer behaves quite linearly up to a threshold at which it no

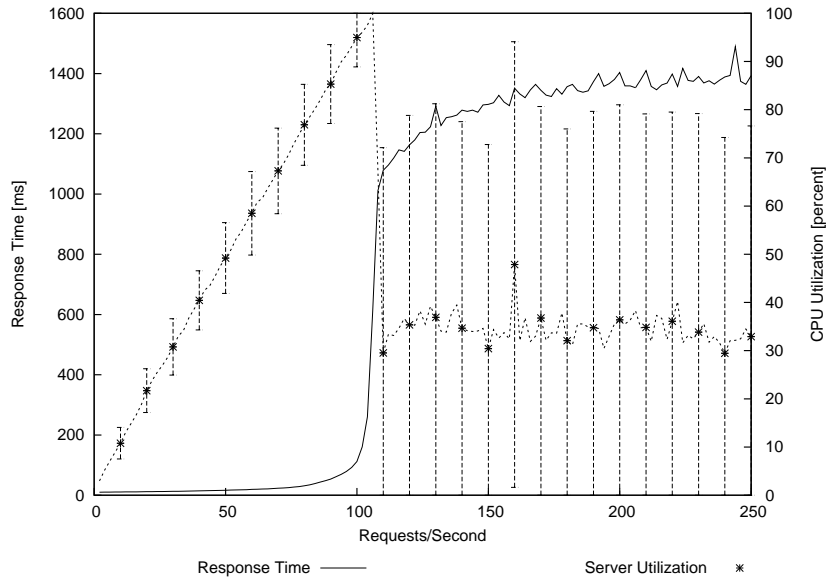


Figure 11: Response time and CPU utilization of a single server as a function of requests per second, using Poisson distribution with exponential inter-arrival times.

longer is able to cope with the resource demands placed upon it. At that point the average service delivery falls to about 50% of the maximum with huge variations (thrashing). At this point one loses essentially all predictability.

Experiments show that, as long as one keeps *all* servers in a server farm underneath this critical threshold, Performance scales approximately linearly when adding identical servers hence we can easily predict the effect of adding more servers as long as load balancing dispatchers themselves behave linearly. Figure 13 shows the addition of servers in four traffic saturation experiments. In the first graph we see that the response time is kept low in all scenarios. This fits with the assumption that even a single server should be capable of handling requests. The next graph shows double this, and all scenarios except the one with a single server are capable of handling the load. This fits the scaling predictions. With four times as much load, in the third graph, four servers just cut it. In the fourth graph we request at six times the load we see that the response rate is high for all scenarios. These results make approximate sense, yet the scaling is not exactly linear; there is some overhead when going from having 1 server to start load balancing between 2 or more servers.

As soon as a server crosses the maximum threshold, response times fall off exponentially. The challenge here is how to prevent this from happening in a data

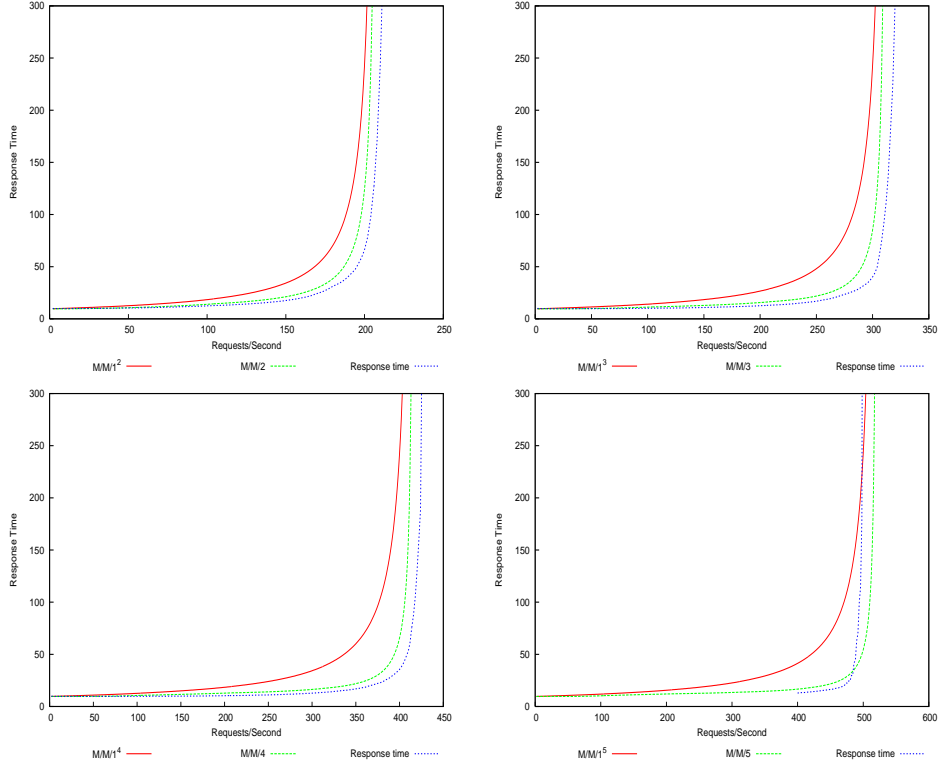


Figure 12: A comparison of $M/M/n$ and $(M/M/1)^n$ queue estimators with response times from a Poisson traffic on real web servers.

centres of inhomogeneous servers with different capacities. Dispatchers use one of a number of different sharing algorithms:

- Round Robin: the classic load sharing method of taking each server in turn, without consideration of their current queue length or latency.
- Least Connections: the dispatcher maintains state over which back end server currently has fewest on-going TCP connections and channels new arrivals to the least connected host.
- Response Time: the dispatcher measures the response time of each server in the back end by regularly testing the time it takes to establish a connection. This has many tunable parameters.

Tests show that, as long as server load is homogeneous, a round robin algorithm is most efficient. However the Least Connections algorithm holds out best against inhomogeneities[9].

10 Configuration and Change Management

A flexible organization is ready to adapt and make changes at any time. Some changes are in response to need and some are preemptive. All change is fraught with risk however. It is therefore considered beneficial to monitor and control the process of change in an organization in such a way that changes are documented and undergo a process of review. Two phrases are used, with some ambiguity, in this connection:

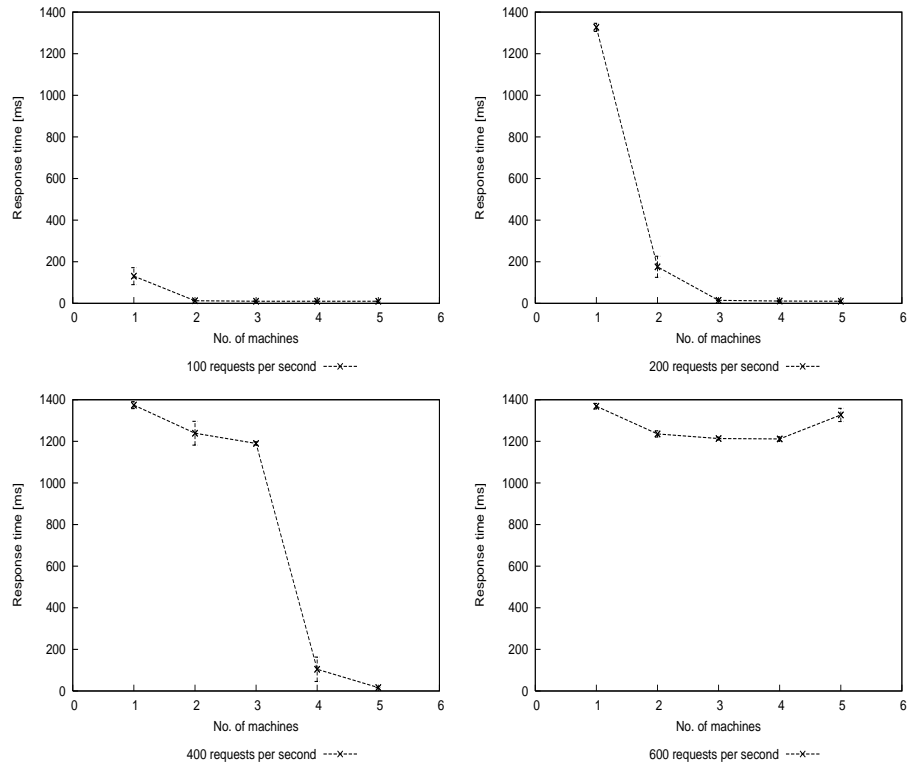


Figure 13: How response rates scale when adding extra servers. For low traffic, adding a new server results in a discontinuous improvement in response time. At very high load, adding a server seems to reduce performance, leading to the upward curve in the last figure as the limitations of the bottleneck dispatcher become apparent.

- Change Management.
- Configuration Management.

Change management is the planning, documentation and implementation of changes in system practices, resources and their configurations. The phrase configuration management invites some confusion, since it is used with two distinct meanings that arise from different cultures. In system administration arising from the world of Unix, configuration management is used to refer to the setup, tuning and maintenance of data that affect the behaviour of computer systems. In software engineering and management (e.g. ITIL and related practices) it refers to the management of a database (CMDB) of software, hardware and components in a system, including their relationships within an organization. This is a much higher level view of “configuration” than is taken in Unix management.

10.1 Revision control

One of the main recommendations about change management is the use of revision control to document changes. We want to track changes in order to

- Be able to reverse changes that had a negative impact.
- Analyze the consequences of changes either successful or unsuccessful.

Some form of documentation is therefore necessary. This is normally done with the aid of a revision control system.

Revision control is most common in software release management, but it can also be used to process documentation or specifications for any kind of change (fig. 14). A revision control system is only a versioning tool for the actual change specification; the actual decisions should also be based on a process. There are many process models for change, including spiral development models, agile development, extreme change, followed by testing and perhaps and regret and reversal, etc. In mission critical systems more care is needed than “try and see”. See for instance the discussions on change management in ITIL[26, 25].

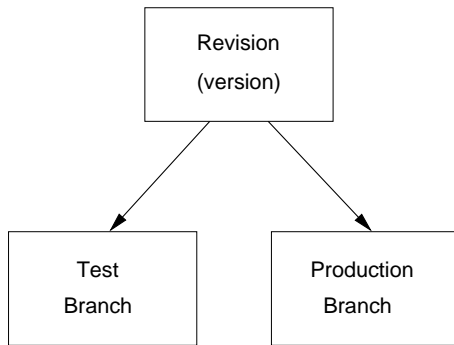


Figure 14: Revision control – changes are tested and only later committed to production.

10.2 “Rollback”

In software engineering one has the notion of going back to an earlier version in the revision control system is a change is evaluated to have negative consequences. Many service managers and data centre engineers would like to have a similar view of live production systems, however there are problems associated with this.

The difference between software code and a live system is that code does not have *runtime operational state*. The behaviour of a system depends in general on both system configuration and learned state that has been acquired through the running of the system. Even if one undoes changes to the configuration of a system, one cannot easily undo operational state (e.g. user sessions in an extended transaction). Thus simply undoing a configuration change will not necessarily return the behaviour of a system to its previous condition. An obvious example of this is: suppose one removes all access control from a system so that it becomes overrun by hackers and viruses etc, simply restoring the access control will not remove these users or viruses from the system.

11 Human resources

In spite of the density of technology in a data centre, humans are key pieces of the service delivery puzzle. Services are increasingly about user-experience. Humans cannot therefore be removed from the system as a whole – one deals with human-computer systems[6]. Several issues can be mentioned:

- *Redundancy of workforce* is an important security in an organization. A single expert who has knowledge about an organization is a *single point of failure* just as much as any piece of equipment. Sickness or even vacation time places the organization at risk if the human services are not available to the organization.

- *Time management* is something that few people do well in any organization. One must resist the temptation to spend all day feeling busy without clearing time to work on something uninterrupted. The state of being busy is a mental state rather than a real affliction. Coping with the demands on a person's time is always a challenge. One has to find the right balance between fire-fighting and building.
- *Incident response procedures* are about making a formal response to a system event that is normally viewed to be detrimental to the system. This is a human issue because it usually involves human judgement. Some responses can be automated, e.g. using a tool like cfengine, but in general more cognition is required. This might involve a fault, a security breach or simply a failure to meet a specification, e.g. a Service Level Agreement (SLA).
- *Procedural documentation* of experience-informed procedures is an essential asset to an organization, because:
 - Past experience is not necessarily available to every data centre engineer.
 - People do not easily remember agreed procedures in a stressful situation or emergency.
 - Consistent behaviour is a comfort to customers and management.

These matters are all a part of managing system predictability.

12 Closing remarks

Data centres are complex systems that are essential components in a holistic view of IT service delivery. They require both intelligence and experience to perfect. There are many issues in data centres that system administrators have to deal with that are not covered in this overview. An excellent introduction to heuristic methods and advice can be found in ref. [23].

The main slogan for this review is simple: well designed systems fail rarely to meet their design targets and when they do so, they do so predictably. They are the result of exceptional planning and long hard experience.

Acknowledgement: I have benefitted from the knowledge and work of several former students and colleagues in writing this chapter: Claudia Eriksen, Gard Undheim and Sven Ingebrigt Ulland (all now of Opera Software) and Tore Møller Jonassen. This work was supported in part by the EC IST-EMANICS Network of Excellence (#26854)

References

- [1] G.M. Amdahl. Validity of a the single processor approach to achieving large scale computer capabilities. In *Proceedings of the AFTPS Spring Joint Computer Conference*, 1967.
- [2] J.H. Bjørnstad and M. Burgess. On the reliability of service level estimators in the data centre. In *Proc. 17th IFIP/IEEE Integrated Management*, volume submitted. Springer, 2006.
- [3] British Standard/International Standard Organization. *BS/ISO17799 Information technology – Code of practice for information security management*, 2000.

- [4] British Standards Institute. *BS15000 IT Service Management*, 2002.
- [5] M. Burgess. *Principles of Network and System Administration*. J. Wiley & Sons, Chichester, 2000.
- [6] M. Burgess. *Analytical Network and System Administration — Managing Human-Computer Systems*. J. Wiley & Sons, Chichester, 2004.
- [7] M. Burgess and G. Canright. Scalability of peer configuration management in partially reliable and ad hoc networks. *Proceedings of the VIII IFIP/IEEE IM conference on network management*, page 293, 2003.
- [8] M. Burgess and G. Canright. Scaling behaviour of peer configuration in logically ad hoc networks. *IEEE eTransactions on Network and Service Management*, 1:1, 2004.
- [9] M. Burgess and G. Undheim. Predictable scaling behaviour in the data centre with multiple application servers. In *Proc. 17th IFIP/IEEE Distributed Systems: Operations and Management (DSOM 2006)*, volume submitted. Springer, 2006.
- [10] Jin Cao, William S. Cleveland, Dong Lin, and Don X. Sun. On the nonstationarity of internet traffic. In *SIGMETRICS '01: Proceedings of the 2001 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, pages 102–112, New York, NY, USA, 2001. ACM Press.
- [11] Peter M. Chen, Edward K. Lee, Garth A. Gibson, Randy H. Katz, and David A. Patterson. Raid: high-performance, reliable secondary storage. *ACM Comput. Surv.*, 26(2):145–185, 1994.
- [12] Hyoung-Kee Choi and John O. Limb. A behavioral model of web traffic. In *ICNP '99: Proceedings of the Seventh Annual International Conference on Network Protocols*, page 327, Washington, DC, USA, 1999. IEEE Computer Society.
- [13] R.B. Cooper. *Stochastic Models*, volume 2 of *Handbooks in Operations Research and Management Science*, chapter Queueing Theory. Elsevier, 1990.
- [14] Mark E. Crovella and Azer Bestavros. Self-similarity in world wide web traffic: evidence and possible causes. *IEEE/ACM Trans. Netw.*, 5(6):835–846, 1997.
- [15] G.R. Ganger, B.L. Worthington, R.Y. Hou, and Y.N. Patt. Disk subsystem load balancing: disk striping vs. conventional data placement. In *Proceeding of the Twenty-Sixth Hawaii International Conference on System Sciences*, volume 1, pages 40–49, 1993.
- [16] G.R. Ganger, B.L. Worthington, R.Y. Hou, and Y.N. Patt. Disk arrays: high-performance, high-reliability storage subsystems. *Computer*, 27(3):30–36, 1994.
- [17] K.I. Hopcroft, E. Jakeman, and J.O. Matthews. Discrete scale-free distributions and associated limit theorems. *Journal of Mathematical Physics*, A37:L635–L642, 2004.
- [18] A. Høyland and M. Rausand. *System Reliability Theory: Models and Statistical Methods*. J. Wiley & Sons, New York, 1994.
- [19] R. Jain. *The art of computer systems performance analysis*. Wiley Interscience, New York, 1991.

- [20] Thomas Karagiannis, Mart Molle, and Michalis Faloutsos. Long-range dependence: Ten years of internet traffic modeling. *IEEE Internet Computing*, 8(5):57–64, 2004.
- [21] A.H Karp and H.P. Flatt. Measuring parallel processor performance. *Communications of the ACM*, 33(5):539–543, 1990.
- [22] Will E. Leland, Murad S. Taqqu, Walter Willinger, and Daniel V. Wilson. On the self-similar nature of ethernet traffic (extended version). *IEEE/ACM Trans. Netw.*, 2(1):1–15, 1994.
- [23] T. Limoncelli and C. Hogan. *The Practice of System and Network Administration*. Addison Wesley, 2003.
- [24] R. Menuet and W.P. Turner. Continuous cooling is required for continuous availability. Technical report, Uptime Institute, 2006.
- [25] Office of Government Commerce, editor. *Best Practice for Service Delivery*. ITIL: The Key to Managing IT Services. The Stationary Office, London, 2000.
- [26] Office of Government Commerce, editor. *Best Practice for Service Support*. ITIL: The Key to Managing IT Services. The Stationary Office, London, 2000.
- [27] Kihong Park, Gitae Kim, and Mark Crovella. On the relationship between file sizes, transport protocols, and self-similar network traffic. In *ICNP '96: Proceedings of the 1996 International Conference on Network Protocols (ICNP '96)*, page 171, Washington, DC, USA, 1996. IEEE Computer Society.
- [28] Vern Paxson and Sally Floyd. Wide area traffic: the failure of poisson modeling. *IEEE/ACM Transactions on Networking*, 3(3):226–244, 1995.
- [29] K. Raatikainen. Symptoms of self-similarity in measured arrival process of ethernet packets to a file server, 1994.
- [30] G.B. Rodosek. Quality aspects in it service management. *IFIP/IEEE 13th International Workshop on Distributed Systems: Operations and Management (DSOM 2002)*, page 82, 2002.
- [31] H. Simitci and D.A. Reed. Adaptive disk striping for parallel input/output. In *16th IEEE Symposium on Mass Storage Systems*, pages 88–102, 1999.
- [32] L.P. Slothouber. A model of web server performance. In *The 5th International World Wide Web Conference, Paris, France*, 1996.
- [33] W.P. Turner, J.H. Seader, and K.G. Brill. Tier classifications define siet infrastructure performance. Technical report, Uptime Institute, 1996, 2001-2006.
- [34] R.D. van der Mei, R. Hariharan, and P.K. Reeser. Web server performance modeling. *Telecommunication Systems*, 16(3 - 4):361–378, March 2001.
- [35] J. Walrand. *Stochastic Models*, volume 2 of *Handbooks in Operations Research and Management Science*, chapter Queueing Networks. Elsevier, 1990.

What Can Web Services Bring To Integrated Management?

Aiko Pras
University of Twente, The Netherlands

Jean-Philippe Martin-Flatin
NetExpert, Switzerland

Since the turn of the millennium, Web services have pervaded the middleware industry, despite their technical limitations, their ongoing standardization, the resulting lack of stable standards, the different meanings of the term *service* to different people, and the fact that marketing forces have blurred the technical reality that hides behind this term. What is so special about Web services and their underlying Service-Oriented Architecture (SOA)?

As a major technology on the software market, Web services deserve to be investigated from an integrated management perspective. What does this middleware technology bring to management applications? Is it yet another way of doing exactly the same thing? Or does it enable management software architects to open up uncharted territories for building a new generation of management solutions? Can it help us manage networks, systems and services with unprecedented flexibility, robustness and/or scalability?

In this chapter, we first present the technical motivation for shifting toward Web services. In Sections 2, 3 and 4, we describe different facets of this middleware technology: its architecture, its protocols, and the main standards that make up the so-called *Web services stack*. In Sections 5 and 6, we show that Web services can be used in integrated management in an evolutionary manner. The example studied here is network monitoring. We propose a fine-grained Web service mapping of SNMP operations and managed objects, and show that the same management tasks can be performed with this new middleware to implement network monitoring. In Section 6, we go back to the rationale behind SOA and compare it with fine-grained Web services. This leads us to propose another perspective, more revolutionary, where services are coarse-grained, wrapping up autonomous programs rather than getting and setting managed objects. We analyze the challenges posed by these coarse-grained Web services to management applications, and the changes that they require in today's management habits. Finally, we give examples showing how management platforms could gradually migrate toward SOA in the future.

1 MOTIVATION

The charm of novelty should not obliterate the fact that it is unwise to change a working solution. Before jumping on the bandwagon of Web services, supposedly the latest and greatest technology in the middleware market, let us review the current state of affairs in integrated management and investigate whether we need a new solution at all.

1.1 COMMAND LINE INTERFACE

Our first observation is that today's management of networks, systems and services still relies to a great extent on protocols and technologies that were developed several decades ago. For instance, the most frequently used management tool still seems to be the Command Line Interface (CLI), which was introduced in the early days of computing (and later networking) to manage computer systems and network devices. In the Internet world, network operators still often rely on this rather crude interface to configure their equipment. The CLI is not only used interactively (in *attended mode*) when Network Operations Center (NOC) staff log into remote systems to manually monitor and modify their operation, but also in *unattended mode* when management scripts automatically connect to remote machines (e.g., using *expect* [47]) to check their configuration and operation and possibly alter them. In large organizations, these scripts are essential to manage and control the Information Technology (IT) infrastructure in a cost-effective and reliable manner.

Unfortunately, different CLIs use different syntaxes: there is no widely adopted standard. Not only do CLIs vary from vendor to vendor, but also from product line to product line for a given vendor, and sometimes even between versions of a given product. Scripts are therefore highly customized and equipment specific, and considerable investments are needed to maintain them as networks, systems and services evolve and come to support new features.

1.2 SNMP PROTOCOL

In the 1970s and early 1980s, when large networks began to be built all over the world, this interoperability issue became increasingly serious. In the 1980s, the International Organization for Standardization (ISO) and the Internet Engineering Task Force (IETF) proposed to address it by defining standard management protocols.

To date, the most successful of these protocols has been the IETF's Simple Network Management Protocol (SNMP), which is accompanied by a standard describing how to specify managed objects—the Structure of Management Information (SMI)—and a document defining standard managed objects organized in a tree structure—the Management Information Base (MIB-II). As the latter defines only a small number of managed objects, additional MIB modules soon appeared.

The first versions of these SNMP standards were developed in the late 1980s and became IETF standards in 1990 (they are now known as SNMPv1 and SMIv1 [85]). Soon afterward, they were implemented in many products and tools, and used on a very large scale, much larger than expected. In view of this success, the original plan to eventually migrate to ISO management protocols was abandoned.

Instead, the IETF decided to create a second version of the SNMP standards. This version should have better information modeling capabilities, improved performance and, most importantly, stronger security.

The work on information modeling progressed slowly but smoothly through the IETF standard track; SMIV2, the new version of the Structure of Management Information, became a proposed standard in 1993, a draft standard in 1996, and a standard in 1999 [56]. In addition, a new Packet Data Unit (PDU) was defined to retrieve bulk data (*get-bulk*).

Unfortunately, the original developers of SNMPv2 disagreed on the security model and failed to find a compromise. In 1994, two competing versions of SNMPv2 were standardized (SNMPv2p and SNMPv2u), which utterly confused the network management market. In 1996, a compromise was standardized (SNMPv2c) [85] with only trivial support for security (“community” string, i.e. the clear-text identification scheme implemented in SNMPv1). SNMPv2c brought several improvements to SNMPv1 and was widely adopted, but it did not address the market’s main expectation: the support for professional-grade security.

In 1996, a new IETF Working Group was created, with different people, to address security issues in SNMP. This resulted in SNMPv3, which progressed slowly through the standard track and eventually became an IETF standard in 2003.

By the time this secure version of SNMP was released, the market had lost confidence in the IETF’s ability to make SNMP evolve in a timely manner to meet its growing demands. To date, two decades after work on SNMP began in 1987, there is still no standard mechanism in SNMP for distributing management across multiple managers in a portable way... As a result, management platforms have to use proprietary “extensions” of SNMP to manage large networks.

In last resort, a new IETF group was formed in 2001, called the Evolution of SNMP (EoS) Working Group, to resume standardization work on SNMP. But it failed to build some momentum and the group was eventually dismantled in 2003, without any achievement.

WAITING FOR A NEW SOLUTION

SNMPv1 was undoubtedly a great success, but it failed to evolve [79]. The frequent use of SNMP to manage today’s networks should not hide the fact that many organizations also use the CLI and *ad hoc* scripts to manage their network infrastructure. This not only defeats the point of an open and standard management protocol, but also prevents the wide use of policy-based management, because CLI-based scripts alter the configuration settings supposedly enforced by Policy Decisions Points (PDPs, i.e. policy managers in IETF jargon). Today, the market is clearly waiting for a new solution for network management.

SNMP never encountered much success in systems and service management. Since its inception, service management has been dominated by proprietary solutions; it still is. The situation was similar in systems management until the early 2000s. Since then, the Web-Based Enterprise Management (WBEM) management architecture of the Distributed Management Task Force (DMTF) has built some momentum and is increasingly adopted, notably in storage area networks. The DMTF has focused mostly on information modeling so far. The communication model of WBEM can be seen as an early version of SOAP.

For large enterprises and corporations, the main problem to date is the integration of network, systems and service management. These businesses generally run several management platforms in parallel, each dedicated to a management plane or a technology (e.g., one for network management, another for systems management, a third one for service management, a fourth one for the trouble-ticket management system, and a fifth one for managing MPLS-based VPNs¹). In these environments, management platforms are either independent (they do not communicate) or integrated in an *ad hoc* manner by systems integrators (e.g., events from different platforms are translated and forwarded to a single event correlator). For many years, the market has been waiting for a technology that would facilitate the integration of these platforms. Researchers have been working on these issues for years, but turning prototypes into reliable and durable products that can be deployed on a very large scale still poses major issues. The market is ripe now for a new technology that could make this integration easier.

1.3 FROM SPECIFIC TO STANDARD TECHNOLOGIES

For many years, the IETF encouraged the engineers involved in its Working Groups to devise a tailor-made solution for each problem. This led to a plethora of protocols. SNMP was one of them: as network bandwidth, Central Processing Unit (CPU) power and memory were precious few resources in the late 1980s, their usage had to be fine-tuned so as not to waste anything.

Since then, the context has changed [53]. First, the market has learned the hard way what it costs to hire, train and retain highly specialized engineers. Second, considerably more network and computing resources are available today: there is no need anymore for saving every possible bit exchanged between distant machines. Third, software production constraints are not the same: in industry, a short time-to-market and reasonable development costs are considerably more important than an efficient use of equipment resources.

For integrated management, the main implications of this evolution are twofold. First, it is now widely accepted that using dedicated protocols for exchanging management information does not make sense anymore: wherever possible, we should reuse existing and general-purpose transfer protocols (pipes) and standard ways of invoking service primitives. Second, instead of using specific technologies for building distributed management applications, we should strive to use standard middleware.

Two platform-independent middleware technologies encountered a large success in the 1990s: the Common Object Request Broker Architecture (CORBA) [67] and Java 2 Enterprise Edition (J2EE) [13]; the latter is also known as Enterprise Java Beans (EJBs). Both made their way to some management platforms for distributing management across several machines, but none of them had a major impact on the integrated management market. Why should Web services do any better?

1. MPLS stands for Multi-Protocol Label Switching, VPN for Virtual Private Network.

1.4 WEB SERVICES

The main difference between Web services and previous types of middleware is that the former support a loose form of integration (*loose coupling*) of the different pieces of a distributed application (*macro-components*¹), whereas the latter rely on their tight integration (*tight coupling*).

The macro-components of a loosely integrated application can work very differently internally: they just need to be able to share interfaces and communicate via these interfaces. What happens behind each interface is assumed to be a local matter. Conversely, the macro-components of a tightly integrated application are internally homogeneous. They are all CORBA objects, or EJBs, etc.

The concept of loose integration is not new. Not only that, but nothing in CORBA (or J2EE) prevents application designers from supporting loose integration, where CORBA objects (or EJBs) serve as interfaces to entire applications; indeed, examples of wrapper CORBA objects (or EJBs) are described in the literature, and some can be found in real-life. But their presence is marginal in the market because this is not how the object-oriented middleware industry has developed.

What is new with Web services is that loose integration is almost mandatory, because Web services are grossly inefficient for engineering the internals of an application. Reengineering a CORBA application with Web services operating at the object level would be absurd: performance would collapse. What makes sense is to wrap up an existing CORBA application with one or several Web service(s), to be able to invoke it from a non-CORBA application.

What is so great about loose integration?

First, customers love it because it preserves their legacy systems, which allows them to cut on reengineering costs. In the past, people working in distributed computing and enterprise application integration had the choice between proprietary middleware (e.g., .NET or DCOM, the Distributed Object Component Model) or interoperable middleware (e.g., CORBA or J2EE). These different types of middleware all assumed a tight integration of software; everything had to be an object: a CORBA object, an EJB, a DCOM component, etc. Using such middleware thus required either application reengineering or brand new developments. Only the wealthiest could afford this on a large scale: stock markets, telecommunication operators and service providers prior to the Internet bubble burst, etc. With Web services, legacy systems are no longer considered a problem that requires an expensive reengineering solution. On the contrary, they are a given, they are part of the solution. When distributed computing and enterprise application integration make use of Web services, not everything needs to be a Web service.

Second, vendors also love loose integration because it allows them to secure lucrative niche markets. Externally, each macro-component of the distributed application looks like a Web service. But internally, it can be partially or entirely proprietary. Interoperability is only assured by interfaces: vendors need not be entirely compatible with other vendors. The

1. In Section 6.2, “Loose Coupling”, we will present examples of macro-components for management applications.

business model promoted by this approach to distributed computing and enterprise application integration is much closer to vendors' wishes than CORBA and J2EE.

In addition to loose integration, Web services present a number of generic advantages that pertain to middleware, eXtensible Markup Language (XML) or the use of well-known technologies [1][53]. For instance, they can be used with many programming languages and many development platforms, and they are included in all major operating systems. Even calling Web services from a Microsoft Excel spreadsheet is easy, and the idea of building simple management scripts within spreadsheets can be appealing [43]. Last but not least, as there are many tools and many skilled developers in this area, implementing Web services-based management applications is usually easier and less expensive than developing SNMP-based applications.

2 INTRODUCTION TO WEB SERVICES

Let us now delve into the technical intricacies of Web services. This technology is defined in a large and growing collection of documents, collectively known as the *Web services stack*, specified by several consortia. In this section, we present the big picture behind Web services, describe the three basic building blocks (SOAP, WSDL and UDDI) and summarize the architecture defined by the World-Wide Web Consortium (W3C). In the next two sections, we will present a selection of standards that are also part of the Web services stack and deal with more advanced features: management, security, transactions, etc.

2.1 COMMUNICATION PATTERNS AND ROLES

Web services implement the Producer-Consumer design pattern [31]. When a Web service consumer binds to a Web service provider (see Figure 1), the consumer sends a request to the provider and later receives a response from this provider. This communication is generally synchronous: even though SOAP messages are one-way, SOAP is often used for synchronous Remote Procedure Calls (RPCs), so the application running the Web service consumer must block and wait until it receives a response [44]. Asynchronous versions of SOAP also exist [73] but are less frequent.

The first difficulty here is that the consumer usually knows the name of the service (e.g., it can be hard-coded in the application), but ignores how to contact the provider for that service. This indirection makes it possible to decouple the concepts of logical service and physical service. For instance, a given service can be supported by provider P1 today and provider P2 tomorrow, in a transparent manner for the applications. But this flexibility comes at a price: it requires a mechanism to enable the consumer to go from the name of a service to the provider for that service.

With Web services, the solution to this problem is illustrated by Figure 1. First, the consumer contacts a service description repository and requests a specific service. Second, the repository returns a handle to the service provider. Third, the consumer can now directly bind to the provider and send a Web service request. Fourth, the provider sends back a Web service response.

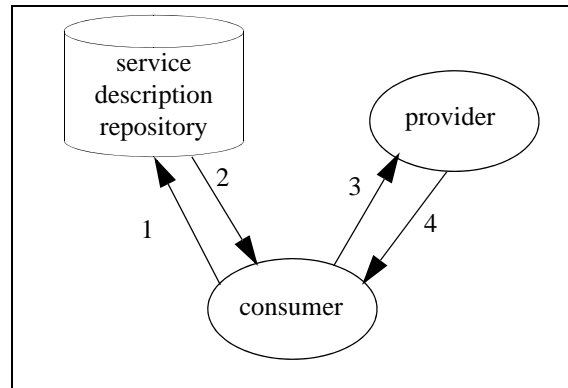


Figure 1: Finding a service offered by just one provider

The second difficulty is that a given service may be supported not by one provider at a time, as before, but by multiple providers in parallel. For instance, different providers may support a variable Quality of Service (QoS) [93], or charge different prices for the same service.

In this case, the consumer typically uses a broker instead of a repository. First, the consumer contacts a trusted broker and requests the best offering for a given service, with parameters that specify what “best” means. Second, the broker builds an ordered list of offers (by contacting repositories, providers or other brokers, or by using locally cached information). Third, the consumer selects an offer and sends a Web service request to that provider. Fourth, the provider sends back a Web service response.

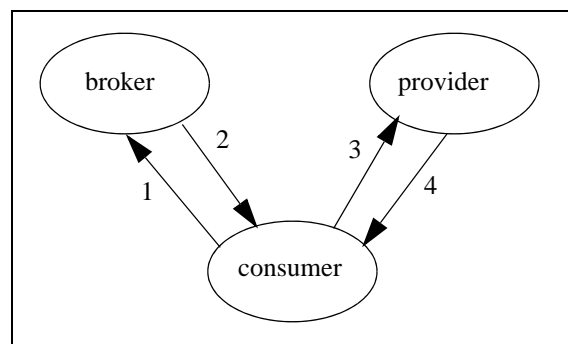


Figure 2: Finding a service offered by multiple providers

2.2 SOAP PROTOCOL

SOAP [32][33] is a protocol that allows Web applications to exchange structured information in a distributed environment. It can be viewed as a simple mechanism for turning service invocations into XML messages, transferring these messages across the network, and translating them into service invocations at the receiving end [1].

SOAP can be used on top of a variety of transfer protocols: HyperText Transfer Protocol (HTTP) [29], Blocks Extensible Exchange Protocol (BEEP) [68], etc. The SOAP binding used by most applications to date specifies how to carry a SOAP message within an HTTP entity-body; with this binding, SOAP can be viewed as a way to structure XML data in an HTTP pipe between two applications running on distant machines.

A SOAP message is an *envelope* that contains a *header* and a *body*. The header is optional and carries metadata; the body is mandatory and includes the actual application payload. Both the header and the body may be split into *blocks* that can be encoded differently. A SOAP message is used for one-way transmission between a *SOAP sender* and a *SOAP receiver*, possibly via *SOAP intermediaries*. Multiple SOAP messages can be combined by applications to support more complex interaction patterns such as request-response. SOAP also specifies how to make RPCs using XML.

In its original definition, the SOAP acronym stood for Simple Object Access Protocol. In the version standardized by the W3C, SOAP is no longer an acronym.

2.3 WEB SERVICES DESCRIPTION LANGUAGE

The Web Services Description Language (WSDL) [16][17] is a standard XML language for describing Web services. Each Web service is specified by a *WSDL description* that separates the description of the abstract functionality offered by a service (*type* and *interface* components) from concrete details of the service description (*binding* and *service* components); the latter defines how and where this functionality is offered. Multiple WSDL descriptions can be published in a single WSDL file, which is sometimes called a *WSDL repository*.

The current version of the standard is WSDL 2.0, released in March 2006. The previous version, WSDL 1.1, was standardized in March 2001. Both are widely used today. These two versions of WSDL are quite different. For instance, the *message* component was made obsolete in WSDL 2.0. The *port type* component in WSDL 1.1 evolved into the *interface* component in WSDL 2.0. The *port* component in WSDL 1.1 was renamed *endpoint* in WSDL 2.0. In Section 5.4, we will present the different components of a WSDL description while we study a detailed example.

If we compare Web services with CORBA, the WSDL language corresponds to the CORBA Interface Definition Language (IDL); a WSDL repository is similar to an Interface Repository in CORBA; applications can discover Web services in a WSDL repository and invoke them dynamically, just as CORBA applications can discover an object interface on the fly and invoke it using the Dynamic Invocation Interface. However, the underlying development paradigm is quite different. CORBA requires a developer to create an interface before implementing clients and servers that match this interface, whereas WSDL descriptions may be provided after the initial creation of the service. Moreover, it is not

necessary to store all WSDL descriptions in a designated repository (whereas it is mandatory in CORBA): the Web service provider may also choose to serve a WSDL description at the physical location where the service is offered.

2.4 UNIVERSAL DESCRIPTION, DISCOVERY AND INTEGRATION

Universal Description, Discovery and Integration (UDDI) [87][88] is a standard technology for looking up Web services in a registry. A UDDI registry offers “a standard mechanism to classify, catalog and manage Web services, so that they can be discovered and consumed” [21]. Here are typical scenarios for using a UDDI registry [21]:

- Find Web services based on an abstract interface definition.
- Determine the security and transport protocols supported by a given Web service.
- Search for services based on keywords.
- Cache technical information about a Web service, and update this cached information at run-time.

Three versions of UDDI have been specified to date. UDDIv1 is now considered historic. The market is migrating from UDDIv2 [6] to UDDIv3 [21]; both are currently used. The main novelties in UDDIv3 include the support for private UDDI registries [1] and registry interaction and versioning [87].

The XML schema that underlies UDDI registries consists of six elements [21]:

- *businessEntity* describes an organization that provides Web services; this information is similar to the yellow pages of a telephone directory: name, description, contact people, etc.;
- *businessService* describes a collection of related Web services offered by an organization described by a *businessEntity*; this information is similar to the taxonomic entries found in the white pages of a telephone directory;
- *bindingTemplate* provides the technical information necessary to use a given Web service advertised in a *businessService*; it includes either the access point (e.g., a Uniform Resource Locator, URL) of this Web service or an indirection mechanism that leads to the access point; conceptually, this information is similar to the green pages of a telephone directory;
- *tModel* (technical model) describes a reusable concept: a protocol used by several Web services, a specification, a namespace, etc.; references to the *tModel*'s that represent these concepts are placed in a *bindingTemplate*; as a result, *tModel*'s can be reused by multiple *bindingTemplate*'s;
- *publisherAssertion* describes a relationship between a *businessEntity* and another *businessEntity*; this is particularly useful when a group of *businessEntity*'s represent a community whose members would like to publish their relationships in a UDDI registry (e.g., corporations with their subsidiaries, or industry consortia with their members);
- *subscription* enables clients to register their interest in receiving information about changes made in a UDDI registry. These changes can be scoped based on preferences provided by the client.

There are well-known problems with UDDI [1][55]: performance, scalability, taxonomies, etc. In practice, this technology is not widely used for publishing and discovering Web services; when it is used, it is often “extended” with proprietary enhancements. Web service discovery is still a very active research area, particularly in the field of semantic Web services (see Section 3.7).

2.5 WEB SERVICES ARCHITECTURE

The Web Services Architecture (WSA) was an attempt by the W3C to “lay the conceptual foundation for establishing interoperable Web services” [9]. This document defines the concept of Web service, architectural models (the message-oriented model, the service-oriented model, the resource-oriented model and the policy model), relationships that are reminiscent of Unified Modeling Language (UML) relationships, and a hodgepodge of concepts grouped under the name “stakeholder’s perspectives”: SOA, discovery, security, Peer to Peer (P2P), reliability, etc.). Management is quickly mentioned in Section 3.9 of [9], but the W3C seems to have paid little attention to this issue thus far.

A more interesting attempt to define an architecture for Web services can be found in Section 5.3 of [1].

3 ADVANCED WEB SERVICES

3.1 STANDARDIZATION

Because interoperability is crucial to Web services, their standardization has been of key importance since their inception. So far, two consortia have been particularly active in this field: the W3C and the Organization for the Advancement of Structured Information Standards (OASIS). More recently, other industrial consortia have worked in this field and produced standards or so-called “best practices”, including the Web Services Interoperability Organization (WS-I), the Open Grid Forum (OGF), the Distributed Management Task Force (DMTF) and Parlay.

3.2 INTEROPERABILITY

In April 2006, WS-I specified the Basic Profile 1.1 [4], a set of implementation guidelines to help people build interoperable Web services. This document contains “clarifications, refinements, interpretations and amplifications of those specifications which promote interoperability”. The main focus is on SOAP messaging and service description. In particular, the guidelines for using SOAP over HTTP are important. A number of Web services standards refer to this Basic Profile.

3.3 COMPOSITION

A *composite Web service* is a group of Web services that collectively offer some functionality. Each Web service taking part in a given composite Web service is known as a *participant*. Each participant can itself be a composite Web service, which can lead to a nested structure of services of arbitrary depth. The process of bundling Web services

together to form a composite Web service is known as *service composition*. A Web service that is not composite is called *atomic* (or *basic* [1]).

There are many ways of specifying composite Web services, expressing constraints between participants, and controlling their synchronization and execution. They are usually grouped into two categories: static and dynamic. Static composition is the simplest form of composition: composite Web services are defined in advance, once and for all.

A generally more useful, but also considerably more complex, form of composition is the dynamic composition of Web services. In this case, Web services are composed at run-time: the participants of a composite Web service are chosen dynamically based on a variety of concerns: availability, load-balancing, cost, QoS, etc. The main challenge here is to build composite Web services semi-automatically or in a fully automated manner. Different techniques have been proposed in the literature [78]; they borrow heavily from other existing and well-established fields (e.g., workflow systems or artificial intelligence planning).

An interesting issue in Web service composition is how to compute the QoS of a composite Web service when we know the QoS offered by each participant Web service [71]. Complex composite services can lead to situations where the error bars become ridiculously large when simple multiplicative techniques are used; reducing these error bars is non-trivial.

Whether a Web service is atomic, statically composed or dynamically composed is transparent to its consumers. Service composition is a key characteristic of Web services: their success in e-business owes a great deal to service composition.

3.4 ORCHESTRATION AND CHOREOGRAPHY

For many years, the terms *orchestration* and *choreography* were used interchangeably in the Web services community. There is still no consensus on their exact meaning. The W3C proposes to distinguish them as follows [34].

ORCHESTRATION

In the realm of Web services, orchestration is about controlling the execution of a composite Web service, viewed as a business process. It specifies how to control, from a central point, the execution and synchronization of all the participant Web services (e.g., by triggering the execution of Web service WS2 when Web service WS1 completes).

Orchestration is usually implemented by an executable process that interacts with all Web services involved. Orchestration languages make it possible to define the order in which the participant Web services should be executed, and to express the relationships between Web services in a workflow manner.

The most famous orchestration language to date is probably WS-BPEL (Web Service Business Process Execution Language) [2], an OASIS draft standard formerly known as BPEL4WS (Business Process Execution Language for Web Services) [22] and already widely used. The main objectives of WS-BPEL are to describe process interfaces for business protocols and define executable process models.

CHOREOGRAPHY

Choreography is a declarative way of defining how participants in a collaboration (e.g., a composite Web service) should work together, from a global viewpoint, to achieve a common business goal. The objective here is to define a common observable behavior by indicating in a contract where information exchanges occur, what rules govern the ordering of messages exchanged between participants, what constraints are imposed on these message exchanges, and when the jointly agreed ordering rules are satisfied [45].

The need for a high-level contract, independent of execution and implementations, stems from the fact that enterprises are often reluctant to delegate control of their business processes to their partners when they engage in collaborations. Choreographies enable partners to agree on the rules of collaboration for a composite Web service without specifying how each participant Web service should work. Each participant is then free to implement its own portion of the choreography as determined by the global view [45]. For instance, one participant may use WS-BPEL for orchestrating business processes, while another participant may use J2EE.

The choreography language standardized by the W3C is called the Web Services Choreography Description Language (WS-CDL) [45]. With this language, it is supposedly easy to determine whether each local implementation is compliant with the global view, without knowing the internals of these local implementations.

3.5 TRANSACTIONS

Classic transactions are short-lived operations that exhibit the four ACID properties: Atomicity, Consistency, Isolation and Durability. These properties are fulfilled by transactional systems using a coordinator and a two-phase protocol called *two-phase commit*. During the first phase, the coordinator contacts each participant in the transaction and asks them to make local changes in a durable manner, so that they can either be rolled back (i.e., cancelled) or committed (i.e., confirmed) later. During the second phase, we have two options. If a failure occurred on any of the participants during phase one, the coordinator sends an abort to each participant and all changes are rolled back locally by the participants; the transaction then fails. Otherwise, the coordinator sends a commit to each participant and all changes are committed locally by the participants; the transaction completes successfully. This two-phase protocol is blocking: once they have completed phase 1 successfully, the participants block until the coordinator sends them a commit (phase 2).

This *modus operandi* is not applicable to Web services [48][70]. First, Web services are not necessarily blocking (e.g., when they rely on an asynchronous implementation of SOAP). Second, in the realm of Web services, transactions are usually long-lived; using the above-mentioned protocol may cause resources to remain unavailable for other transactions over extended periods of time, thereby impeding concurrency. Third, when composite Web services are used across multiple organizations, security policies often prevent external entities from hard-locking a local database (allowing it would open the door to denial-of-service attacks, for instance). Hence another mechanism is needed.

Several techniques were devised in the past [48] to release early the resources allocated by a long-lived transaction, and allow other transactions to run in the meantime. In case of

subsequent failure, compensation mechanisms make it possible to bring the transactional system to the desired state (e.g., a payment by credit card can be reimbursed). However, these compensations do not guarantee all ACID properties. This is the approach generally adopted for Web services [1]. Another possibility is to subdivide long-lived transactions into independent short-lived transactions that can run independently as classic atomic transactions.

Detailing the support for transactions in a Web service environment would require an entire book. The OASIS consortium is the main standardization actor in this arena. It adopted an approach similar to Papazoglou's [70], where transactions are part of a bigger framework that defines business protocols, coordination protocols, transactions, orchestration of business processes, choreography, etc. Several building blocks have already been standardized: the business transaction protocol (WS-BTP), atomic transactions (WS-Atomic-Transaction), business activities (WS-BusinessActivity), coordination protocols (WS-Coordination), transactions (WS-Transaction), etc. These standards are still evolving and should not be considered stable yet. Readers interested in learning more about transactions in the context of Web services are referred to the Web site of the OASIS Web Services Transaction Technical Committee [65].

3.6 SECURITY

WS-Security specifies how to secure the SOAP protocol. By supporting end-to-end application-level security, it nicely complements HTTPS, the secure version of HTTP, which can only secure communication on a hop-by-hop basis (with HTTPS, each intermediary can decrypt and re-encrypt the HTTP message) [1]. Whereas the entire payload is encrypted with HTTPS, WS-Security makes it possible to encrypt only one block of the SOAP body (e.g., bank details).

WS-Security was standardized by OASIS in 2004 (version 1.0) and upgraded in 2006 (version 1.1). It consists of a main document, known as SOAP Message Security, and several companion documents.

SOAP Message Security 1.1 [60] specifies SOAP extensions for sending security tokens as part of a SOAP message, guaranteeing message integrity (payload) and ensuring message confidentiality. This specification makes it possible to secure Web services with various security models: Public Key Infrastructure (PKI), Kerberos, Secure Sockets Layer (SSL), etc. *Per se*, it does not provide a complete security solution for Web services; instead, "it can be used in conjunction with other Web service extensions and higher-level application-specific protocols to accommodate a wide variety of security models and security technologies" [60].

In version 1.1, there are six companion documents: Username Token Profile (to identify the requestor by using a username, and optionally authenticate it with a password), X.509 Certificate Token Profile (to support X.509 certificates), the SAML Token Profile (to use Security Assertion Markup Language assertions as security tokens in SOAP headers), the Kerberos Token Profile (to use Kerberos tokens and tickets), the REL Token Profile (to use the ISO Rights Expression Language for licenses), and the SwA Profile (to secure SOAP messages with Attachments).

3.7 SEMANTIC WEB SERVICES

In 2001, Berners-Lee launched the Semantic Web with a powerful statement that has become famous: “The Semantic Web will bring structure to the meaningful content of Web pages” [7]. The idea was to add more metadata to Web pages (using XML) to enable software agents to parse them automatically and “understand” their hierarchically structured contents (containment hierarchy), as opposed to seeing only flat structures and keywords out of context. The main building blocks defined so far for the Semantic Web are the Resource Description Framework (RDF) [51] and the Web Ontology Language (OWL) [58].

In the realm of Web services, this vision translated into the concept of semantic Web services [59]. Whereas Web services focus mostly on interoperability, semantic Web services complement interoperability with automation, dynamic service discovery and dynamic service matching. Semantic Web services address the variety of representations (e.g., different names for the same concept, or different signatures for semantically equivalent operations) by using automated or semi-automated ontology-matching mechanisms [69]. This makes it possible, for instance, to compose Web services at run-time with increased flexibility. OWL-S (Web Ontology Language for Services) [52] is probably the most famous language to date for specifying semantic Web services.

4 WEB SERVICES FOR MANAGEMENT, MANAGEMENT OF WEB SERVICES

In this section, we review the main standards pertaining to (i) the use of Web services for managing networks, systems and services, and (ii) the management of Web services.

4.1 OASIS: WEB SERVICES FOR DISTRIBUTED MANAGEMENT (WSDM)

Unlike the W3C, OASIS has already paid much attention to management issues in the Web services arena. In 2005, the Web Services Distributed Management (WSDM) Technical Committee [63] standardized version 1.0 of a group of specifications collectively called WSDM. In August 2006, version 1.1 was released. To date, these standards are probably the most relevant to the integrated management community as far as Web services are concerned. Let us describe the main ones.

Management Using Web Services (MUWS) [11][12] is a two-part series of specifications that define how to use Web services in distributed systems management. First, MUWS defines a terminology and roles. As prescribed by WSDL and WS-Addressing [10], a Web service is viewed as an aggregate of *endpoints*. Each endpoint binds a Web service interface (described by a WSDL `portType` element) to an address (URL). Each interface describes the messages that can be exchanged and their format [91]. A *manageable resource* is a managed entity (e.g., a network device or a system). A *manageability endpoint* is bound to a manageable resource and can be accessed by a Web service consumer. It acts as a gateway between Web services and a given manageable resource. It can interact with the manageable resource either directly or via an agent (e.g., an SNMP agent). A *manageability consumer* is a Web service consumer, a Web-enabled managing entity (e.g., a management application running on a remote network management system). After discovering Web

service endpoints, it can exchange SOAP messages with these endpoints to request management information (monitoring in polling mode), to subscribe to events (e.g., if a PC needs to report that the temperature of its mother board is too high), or to configure the manageable resource associated with a manageability endpoint [11].

Second, MUWS defines some rules that govern communication between manageability endpoints and consumers (i.e., between managing and managed entities). To discover the manageability endpoint that enables a given manageable resource to be accessed by a Web service, a manageability consumer must first retrieve an Endpoint Reference [10]; next, it can optionally retrieve a WSDL file, a policy, etc. Once the discovery phase is over, the manageability consumer (managing entity) can exchange messages with a manageability endpoint (managed entity) by using information found in the Endpoint Reference [11].

Third, MUWS specifies messaging formats to achieve interoperability among multiple implementations of MUWS (e.g., when management is distributed hierarchically over different domains [82] controlled by different managing entities) [12].

Management of Web Services (MOWS) [91] specifies how to manage Web services. It defines the concept of Manageability Reference (which enables a manageability consumer to discover all the manageability endpoints of a manageable resource), and a list of manageability capabilities (metrics for measuring the use and performance of Web services, operational state of a managed Web service, processing state of a request to a managed Web service, etc.) [91].

4.2 DMTF: WS-MANAGEMENT

WS-Management [23] was standardized by the DMTF in April 2006 to promote Web services-based interoperability between management applications and managed resources. It is independent of the WBEM management architecture and the Common Information Model (CIM).

First, WS-Management defines a terminology and roles. A *managed resource* is a managed entity (e.g., a PC or a service). A *resource class* is the information model of a managed resource; it defines the representation of management operations and properties. A *resource instance* is an instance of a resource class (e.g., a CIM object).

Second, WS-Management specifies mechanisms (i) to get, put, create, and delete resource instances; (ii) to enumerate the contents of containers and collections (e.g., logs); (iii) to subscribe to events sent by managed resources; and (iv) to execute specific management methods with strongly typed parameters [23].

The approach promoted by WS-Management corresponds to what we call fine-grained Web services in this chapter. In Section 5, we will see Web services (this time in the SNMP world) that similarly perform a `get` on an Object Identifier (OID), i.e. a managed object.

The relationship between WSDM and WS-Management is not crystal clear. They do overlap but the scope of WSDM is much wider than that of WS-Management, so they are not really competitors (at least not for now). It seems that WS-Management could eventually provide a migration path for exchanging CIM objects between managing and managed entities, from the DMTF's pre-Web services and pre-SOAP "CIM operations over HTTP" to a Web services-based communication model for WBEM. In this case, WS-Management

would only target the WBEM community, not the Web services community at large. This is however mere speculation and remains to be seen.

4.3 OASIS: WS-RESOURCE

In 2001, the Global Grid Forum (GGF), now known as the OGF, began working on the modeling of Grid resources. This led to the creation of the Open Grid Services Infrastructure (OGSI) Working Group. In 2002 and 2003, the concepts of Grid services and Web services gradually converged, as the specifics of Grids were separated from features that belonged in mainstream standardization. A number of GGF group members got involved in W3C and OASIS groups, and several ideas that had emerged within the GGF percolated to these groups. GGF focused on its core business (Grid-specific activities) and more generic standardization efforts moved to other standards bodies.

In early 2004, a group of vendors issued a draft specification for modelling and accessing persistent resources using Web services: Web Services Resource Framework (WSRF) version 1.0. It was inspired by OGSI and played an important role in the convergence of Web and Grid services. Shortly afterward, this standardization effort was transferred to OASIS, which created the WSRF Technical Committee [64] to deal with it. This led to the phasing out of OGSI and its replacement by a series of documents collectively known as WSRF. In April 2006, version 1.2 of WSRF was released by OASIS. It includes WS-Resource [30], WS-ResourceProperties, WS-ResourceLifetime, WS-ServiceGroup and WS-BaseFaults (for more details, see Banks [5]).

4.4 PARLAY-X

The multi-vendor consortium called Parlay has defined a number of open service-provisioning Application Programming Interfaces (APIs) to control various aspects of Intelligent Networks. At the same time the Open Service Access (OSA) group of the 3rd Generation Partnership Project (3GPP) was working on similar APIs that allowed applications to control a Universal Mobile Telecommunications System (UMTS) network. These two groups joined their activities and now common Parlay/OSA API standards exist. These APIs require detailed knowledge of the Intelligent Networks, however, which means that only a limited number of experts are able to use these APIs. The Parlay group therefore defined some easier to use APIs, at a higher level of abstraction. These APIs, called Parlay-X, are defined in terms of Web services. Examples of such services include “connect A to B”, “give status of X (on/off)”, “send SMS¹”, “give location of mobile handset” and “recharge pre-paid card”. The users of a Parlay-X server do not need to know the details of the underlying network; the Parlay-X server translates the Web services calls into Parlay/OSA calls that are far more difficult to understand. It is interesting to note that Web services technologies were applied by the Parlay group long before the IETF or the Internet Research Task Force (IRTF) even started to think about these technologies.

1. Short Message System.

4.5 IETF AND IRTF-NMRG

In the summer of 2002, the Internet Architecture Board (IAB) organized a one-off Network Management Workshop to discuss future technologies for Internet management. One of the conclusions of this workshop was that time had come to investigate alternative network management technologies, in particularly those that take advantage of the XML technology [79].

Web services are a specific form of XML technology. The 11th meeting of the IRTF Network Management Research Group (NMRG) [38], which was organized three months after the IAB workshop, therefore discussed the possible shift toward XML and Web services-based management. Although many questions were raised during that meeting, the most interesting question was that of performance; several attendees expressed their concern that the anticipated high demands of Web services on network and agent resources would hinder, or even prohibit, the application of this technology in the field of network management. At that time, no studies were known that compared the performance of Web services to that of SNMP. Most attendees therefore preferred to take a simpler approach not based on Web services (e.g., JUNOScript for Juniper). The idea behind JUNOScript is to provide management applications access to the agent's management data, using a lightweight RPC mechanism encoded in XML. As opposed to SNMP, JUNOScript uses a connection-oriented transport mechanism (e.g., `ssh` or `telnet`). An advantage of JUNOScript is that related management interactions can be grouped into sessions, which makes locking and recovery relatively simple. In addition, contrary to SNMP, management information is not limited in size and can be exchanged reliably. JUNOScript provided the basis for a new IETF Working Group called NetConf. Details about NetConf can be found in another chapter of this book.

Within the IRTF-NMRG, several researchers continued to study how to use Web services in network, systems and service management. Some of the results of this work are presented in the next section.

5 FINE-GRAINED WEB SERVICES FOR INTEGRATED MANAGEMENT

At first sight, an intuitive approach to introduce Web services in network management would be to map existing management operations (e.g., the SNMP operations `get`, `set` and `inform`) onto Web services. Is this efficient? Is this the way Web services should be used?

5.1 FOUR APPROACHES

Mapping the current habits in SNMP-based network management onto Web services leads to four possible approaches, depending on the genericity and the transparency chosen [83]. *Genericity* characterizes whether the Web service is generic (e.g., a Web service that performs a `get` operation for an OID passed as parameter) or specific to an OID (e.g., `getIfInOctets`). *Transparency* is related to the parameters of the Web service, which can be either defined at the WSDL level or kept as an opaque string that is defined as part of a

higher-level XML schema; this will be referred to as Web services that have either transparent or non-transparent parameters. The four resulting forms of Web service are depicted in Figure 3.

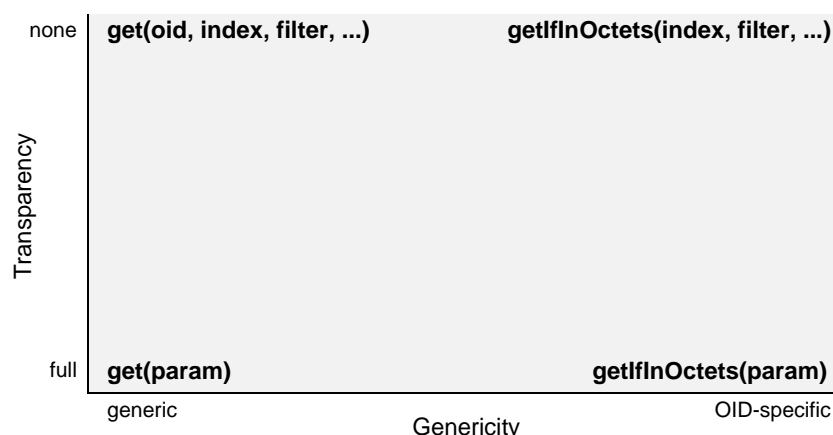


Figure 3: Fine-grained Web services: transparency vs. genericity

5.2 PARAMETER TRANSPARENCY

Web service operations are defined as part of the abstract interface of WSDL descriptions. An operation may define the exchange of a single message, such as a `trap`, or multiple messages, such as `getRequest` and `getResponse`. For each message, the parameters are defined in so-called `<part>` elements. One approach consists in passing multiple parameters and defining the syntax of each parameter in a separate `<part>` element. An example of this is a `getRequest` message, which defines three parameters: `oid` (object identifier), `index` and `filter`. This example of non-transparent parameters is showed in Figure 4; note that for the sake of simplicity, all parts are of type `string` in this example.

Another approach is to define just a single parameter, and leave the interpretation of this parameter to the (manager-agent) application. In this case, only a single `<part>` element is needed. The syntax of the parameter is of type `string`, which means that the parameter is transparently conveyed by the WSDL layer and no checking is performed at that layer (see Figure 5).

An advantage of having parameter transparency is that a clear separation is made between the management information and the protocol that exchanges this information. In this case, it is possible to change the structure of the management information without altering the definition of management operations. From a standardization perspective, this is a good selling point. However, this is not a mandatory feature as WSDL supports other constructs (e.g., the `import` statement) to divide a specification over several documents, which can then be processed independently along the standardization track.

In case of transparent parameters, the application can exchange management data in the form of XML documents. In this case, an XML parser is needed to extract the parameters

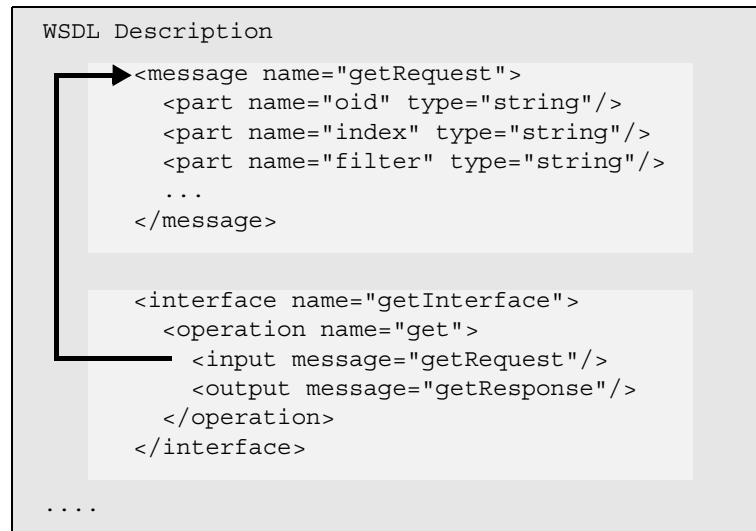


Figure 4: Operation with non-transparent parameters

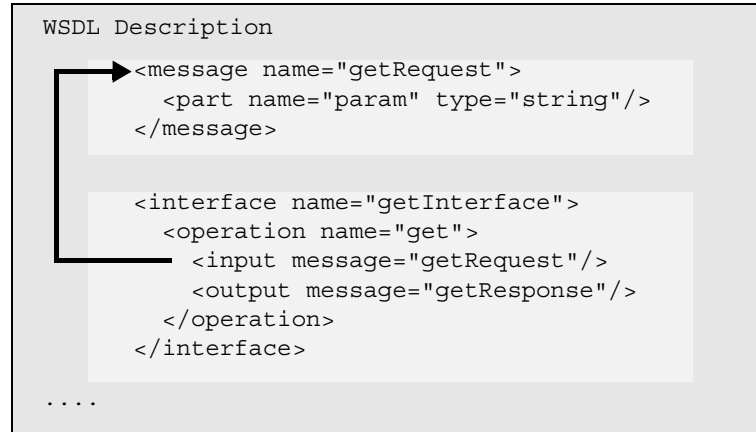


Figure 5: Operation with transparent parameters

from the document. When the application receives such a document, XPath [20] expressions may be used to select the required information. On the manager side, special functionality is needed to create such XML documents. As specific parsing is needed in the application, transparent parameters should only be used by experienced users who need this flexibility. For a PC user, in his home environment, who wants to include some management information in an Excel spreadsheet, this is too complicated. In such a case, a simple approach is required in which the user does not need to parse XML documents with tools like XPath and XQuery. Instead, parsing and checking should be performed at the WSDL layer, thus by the spreadsheet itself. Non-transparent parameters are therefore better suited for simple management applications, as found in home or small-enterprise environments.

5.3 GENERICITY

Another approach is to vary the genericity of operations, as illustrated by this example. Assume a managed system provides host-specific information such as `SysLocation` and `SysUptime`, as well as statistical information for the network interfaces such as `IfInOctets`, `IfOutOctets`, `IfInErrors` and `IfOutErrors`. As the system can have multiple interfaces, the four interface-related OIDs can be provided multiple times. Figure 6 shows this information in the form of a containment tree.

To retrieve this information, management operations need to be defined. If these operations are defined at a low level of genericity, a dedicated operation is required for each variable. The resulting operations then look like `getSysUptime` or `getIfOutOctets` (see Figure 6). Note that the operations for retrieving network interface information, need to have a parameter supplied to identify the interface.

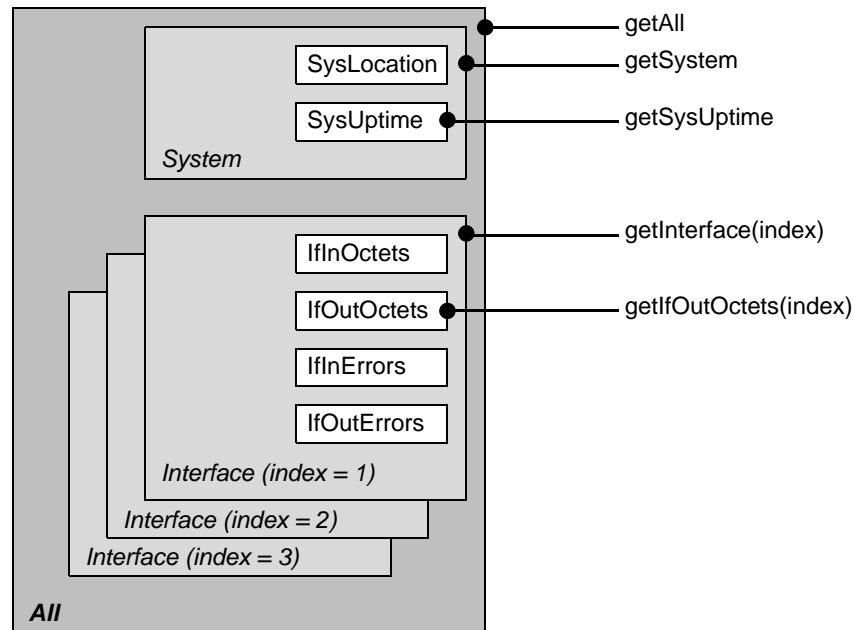


Figure 6: Containment diagram

Management operations can also be defined at a higher level of genericity. For example, a `getSystem` operation can be defined to retrieve `SysLocation` as well as `SysUptime`, and a `getInterface(index)` operation to retrieve all management information for a specific interface. If we push this rationale even further, a single `getAll` operation can retrieve all information contained in the managed system.

An important characteristic of this approach is that the naming of the operations precisely defines the functionality. Therefore it is very easy for the manager to determine which operation to call to retrieve certain information. Note that the number of parameters associated with each operation may be small: no OID is needed because the name of the

operation already identifies the object. Only in case multiple instances of the same object class exist (e.g., the `Interface` object in Figure 6) does the manager have to provide an instance identifier.

Like in other management architectures such as OSI Management (the Open Systems Interconnection management architecture devised by the ISO) [35] or TMN (the Telecommunications Management Network devised by the International Telecommunication Union, ITU) [35], the containment hierarchy can be presented as a tree (see Figure 7). Fine-grained operations can be used to retrieve the leaves of the tree, and coarse-grained operations to retrieve the leaves as well as their higher-level nodes. In SNMP, conversely, only the leaves of the MIB trees are accessible. In general, however, it is not possible to select in a single operation objects from different trees; if the manager wants to retrieve, for example, the OIDs `System` and `IfInErrors`, two separate SNMP operations are needed.

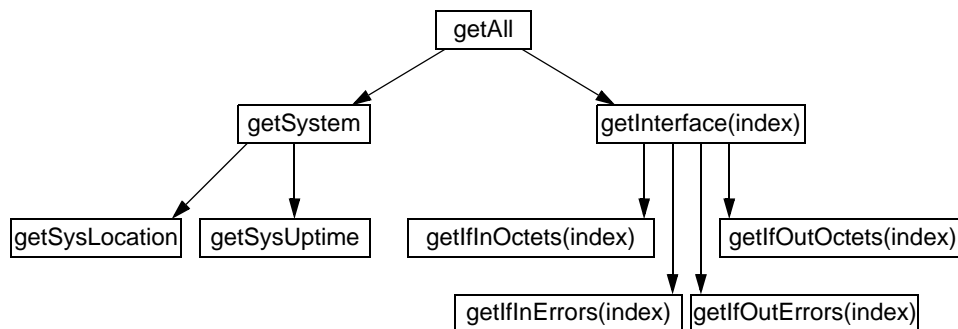


Figure 7: Containment tree

To restrict an operation to specific objects, it is possible to use Web services in combination with filtering. Theoretically, filtering allows usage of `getAll` to retrieve every possible variable, provided that an appropriate filter (and possible index) is given. In combination with filtering, `getAll` can thus be considered an extreme form of genericity, useful for all kinds of task. In practice, however, filtering can be expensive. When it is performed on the agent side, it may consume too many CPU cycles and memory. When filtering is performed on the manager side, bandwidth may be wasted and the management platform may become overloaded. Although filtering is very flexible and powerful, it should be used with care.

5.4 EXAMPLE OF WEB SERVICES-BASED MANAGEMENT

To illustrate the use of Web services for integrated management, let us now study an example of Web services-based network monitoring system. The example is about the SNMP Interface table (`ifTable`), which is part of the Interfaces Group MIB (`IF-MIB`) [57]. The `ifTable` provides information on the operation and use of all network interfaces available in the managed entity. These interfaces can be physical interfaces (e.g., Ethernet and Wireless Local Area Network cards) or virtual interfaces (e.g., tunnels or the loop-back interface). For each interface, the table includes a separate row; the number of rows is therefore equal to the number of interfaces. The table consists of 22 columns and includes

information such as the interface index (`ifIndex`), the interface description (`ifDescr`), the interface type (`ifType`), etc. Figure 8 shows a summary of this table.

	<code>ifIndex</code>	<code>ifDescr</code>	<code>ifType</code>	<code>ifMtu</code>	<code>ifSpeed</code>		<code>ifOutDiscards</code>	<code>ifOutErrors</code>	<code>ifOutQLen</code>	<code>ifSpecific</code>
Interface 1						---				
Interface 2						---				
Interface 3						---				

Figure 8: Interface Table

Before we can come up with a WSDL description for retrieving the Interface table, we must choose the genericity of the WSDL operations. Three levels of genericity are possible, ranging from high genericity to low genericity.

At the highest level of genericity, a single operation can be defined to retrieve all objects of the `ifTable`. We call this Web service operation `getIfTable`.

Instead of retrieving the entire `ifTable` in a single call, it is also possible to define Web service operations that retrieve only a single row or a single column. The operation that retrieves a single row of the Interface table is called `getIfRow`. This operation gets all information related to a single network interface and requires as parameter the identifier of that interface; a possible choice for this parameter is the value of `ifIndex`. The operation that retrieves a single column is called `getIfColumn`. To identify which column should be retrieved, a parameter is needed that can take values such as `ifIndex`, `ifDescr`, `ifType`, etc. Alternatively, it is possible to define separate operations per column; in that case, the resulting operations would be `getIfIndex`, `getIfDescr`, `getIfType`, etc.

Finally, at the lowest level of genericity, separate operations are defined to retrieve each individual `ifTable` object. This approach is somehow comparable to the approach used by the SNMP `get` operation (although the latter allows multiple objects to be retrieved in a single message). Depending on the required parameter transparency, again two choices are possible. The first possibility is to have a generic `getIfCell` operation; this operation requires as input parameters an interface identifier to select the row (interface), as well as a parameter to select the column (e.g., `ifIndex`, `ifDescr` or `ifType`). The second possibility is to have special operations for each column; these operations require as input parameter the identifier of the required interface. An example of such an operation is `getIfType(interface1)`.

Figure 9 summarizes the choices that can be made to retrieve table information; they range from high genericity (`getIfTable`) to low genericity (`getIfCell`). Note that this figure does not show the choices that are possible with respect to parameter transparency.

Let us now discuss the main parts of the WSDL descriptions of these operations. These descriptions are a mix of WSDL 2.0 and WSDL 1.1, as WSDL was not yet finalized when

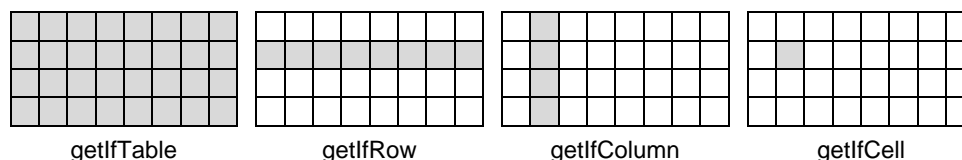


Figure 9: Different choices to retrieve table objects

this work was performed. In the gSOAP toolkit that we used [25], the WSDL descriptions consist of the following (container) elements:

- The `<types>` element (WSDL 2.0), to define new data types.
- The `<message>` elements (WSDL 1.1), to specify the data that belongs to each message.
- The `<interface>` element (WSDL 2.0), to combine one or more messages and ease the generation of programming stubs.
- The `<binding>` element (WSDL 2.0), to associate the interface element with a transport protocol.
- The `<service>` element (WSDL 2.0), to associate the interface element with a web address (URL).

THE `<TYPES>` ELEMENT

XML supports two types for defining elements: `<simpleType>` and `<complexType>`. A `<simpleType>` element contains a single value of a predefined form (e.g., an integer or a string). A `<complexType>` element groups other elements (e.g., an address element can contain street, postal code, and city sub-elements).

The WSDL descriptions of our examples use `<complexType>` elements to group certain objects within the `ifTable`. Which object are taken together depends on the genericity of the Web service. For example, all columns of the `ifTable` can be grouped together, yielding a single `<complexType>` that can be used to retrieve entire `ifTable` rows. This complex type is called `ifEntry` in our example (see Figure 10), and can be used by the `getIfTable` as well as `getIfRow` operations. Note that for the `getIfRow` operation, the `<sequence>` element should be left out, as only one row can be retrieved at a time.

THE `<MESSAGE>` ELEMENTS

The `<message>` elements are used in WSDL 1.1 to describe the information that is exchanged between a Web service producer and its consumers. There are `<message>` elements for request (input) and response (output) messages. These messages consist of zero or more `<part>` elements. In request messages, the `<part>` elements represent the parameters of the Web service. In response messages, these elements describe the response data.

The `getIfTable` Web service supports a single operation: retrieving the complete table. Figure 11 shows the `<message>` elements for this operation. The figure shows a request message that contains a “community” string; this string is used for authentication purposes in SNMPv1 or SNMPv2c. The response message contains an element of type `ifEntry`, which was defined in Figure 10, as well as an integer representing the number of table rows.

```

<types>
  <complexType name="ifEntry">
    <sequence>
      <element name="ifIndex" type="xsd:unsignedInt"/>
      <element name="ifDescr" type="xsd:string"/>
      <element name="ifType" type="xsd:unsignedInt"/>
      <element name="ifMtu" type="xsd:unsignedInt"/>
      <element name="ifSpeed" type="xsd:unsignedInt"/>
      <element name="ifPhysAddress" type="xsd:string"/>
      <element name="ifAdminStatus" type="xsd:unsignedInt"/>
      <element name="ifOperStatus" type="xsd:unsignedInt"/>
      <element name="ifInOctets" type="xsd:unsignedInt"/>
      <element name="ifInUcastPkts" type="xsd:unsignedInt"/>
      <element name="ifInDiscards" type="xsd:unsignedInt"/>
      <element name="ifInErrors" type="xsd:unsignedInt"/>
      <element name="ifOutOctets" type="xsd:unsignedInt"/>
      <element name="ifOutUcastPkts" type="xsd:unsignedInt"/>
      <element name="ifOutDiscards" type="xsd:unsignedInt"/>
      <element name="ifOutErrors" type="xsd:unsignedInt"/>
      <element name="ifOutQLen" type="xsd:unsignedInt"/>
      <element name="ifSpecific" type="xsd:string"/>
    </sequence>
  </complexType>
  ...
</types>

```

Figure 10: The ifEntry type

```

<message name="getIfTableRequest">
  <part name="community" type="xsd:string"/>
</message>

<message name="getIfTableResponse">
  <part name="sizeTable" type="xsd:int"/>
  <part name="ifEntry" type="utMon:ifEntry"/>
</message>

```

Figure 11: Message definitions for getIfTable

All other Web services in this section support multiple operations. The `getIfRow` Web service can be defined to support two operations: retrieve a specified row, and retrieve a list of valid row index numbers (which correspond to the network interfaces in the agent system). The `getIfColumn` and `getIfCell` Web services can be defined in terms of operations (one per `ifTable` column).

Figure 12 shows the messages for two of the operations used by the `getIfCell` Web service. The request messages have an index element for referencing the correct cell.

```
<message name="getIfIndexRequest">
  <part name="index" type="xsd:unsignedInt"/>
  <part name="community" type="xsd:string"/>
</message>

<message name="getIfIndexResponse">
  <part name="ifIndex" type="xsd:unsignedInt"/>
</message>

<message name="getIfDescrRequest">
  <part name="index" type="xsd:unsignedInt"/>
  <part name="community" type="xsd:string"/>
</message>

<message name="getIfDescrResponse">
  <part name="ifDescr" type="xsd:string"/>
</message>
```

Figure 12: Message definitions for getIfCell

THE <INTERFACE> ELEMENT

An <interface> element defines the operations that are supported by the Web service. Each operation consists of one or more messages; the <interface> element therefore groups the previously defined <message> elements into <operation> elements. In addition, the <interface> element may define <description> elements. In general, four types of operation exist: *one way*, *request-response*, *solicit response* and *notification*. In our example, they are all of the *request-response* type. Figure 13 shows the <interface> element for the `getIfTable` Web service.

```
<interface name="getIfTableServicePortType">
  <operation name="getIfTable">
    <documentation>function utMon__getIfTable</documentation>
    <input message="tns:getIfTableRequest"/>
    <output message="tns:getIfTableResponse"/>
  </operation>
</interface>
```

Figure 13: Interface definition for `getIfTable`

The other Web services support multiple operations. Figure 14 shows part of the <interface> element for the `getIfColumn` Web service.

```
<interface name="getIfColumnServiceInterface">
  <operation name="getIfIndex">
    <documentation>function utMon__getIfIndex</documentation>
    <input message="tns:getIfIndexRequest"/>
    <output message="tns:uIntArray"/>
  </operation>
  <operation name="getIfDescr">
    <documentation>function utMon__getIfDescr</documentation>
    <input message="tns:getIfDescrRequest"/>
    <output message="tns:stringArray"/>
  </operation>
  <operation name="getIfType">
    <documentation>function utMon__getIfType</documentation>
    <input message="tns:getIfTypeRequest"/>
    <output message="tns:uIntArray"/>
  </operation>
  ...
</interface>
```

Figure 14: Interface definition for `getIfColumn`

THE <BINDING> ELEMENT

A <binding> element specifies which protocol is used to transport the Web service information, and how this information is encoded. Similar to the <interface> element, it also includes the operations that are supported by the Web service, as well as the request and response messages associated with each operation. In principle, there are many choices for

the transport protocol; in practice, SOAP over HTTP is by far the most popular choice. Figure 15 shows part of the <binding> element for the `getIfColumn` Web service.

```
<binding name="getIfColumnServiceBinding"
        type="tns:getIfColumnServiceInterface">
  <SOAP:binding style="rpc"
    transport="http://schemas.xmlsoap.org/soap/http"/>

  <operation name="getIfIndex">
    <SOAP:operation soapAction=""/>
    <input>
      <SOAP:body use="encoded" namespace="urn:utMon"
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding"/>
    </input>
    <output>
      <SOAP:body use="encoded" namespace="urn:utMon"
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding"/>
    </output>
  </operation>

  <operation name="getIfDescr">
    <SOAP:operation soapAction=""/>
    <input>
      <SOAP:body use="encoded" namespace="urn:utMon"
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding"/>
    </input>
    <output>
      <SOAP:body use="encoded" namespace="urn:utMon"
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding"/>
    </output>
  </operation>
  ...
</binding>
```

Figure 15: Binding definition for `getIfColumn`

THE <SERVICE> ELEMENT

The <service> element is used to give the Web service a name and specify its location (a URL). Figure 16 shows this element for the example of the `getIfRow` Web service. The name is `getIfRowService`, the location is `http://yourhost.com/` and the transport protocol is defined by the `getIfRowServiceBinding`.

```
<service name="getIfRowService">
  <documentation>getIfRow service</documentation>
  <endpoint name="getIfRowService"
    binding="tns:getIfRowServiceBinding">
    <SOAP:address location="http://yourhost.com"/>
  </endpoint>
</service>
```

Figure 16: Service element definition for `getIfRow`

5.5 PERFORMANCE

One of the main arguments against using Web services in network management is that the performance of Web services is inferior to that of SNMP. We show in this section that this argument is invalid. Although there are scenarios in which SNMP provides better performance, there are other scenarios in which Web services perform better. In general, the following statements can be made:

- In case a single managed object should be retrieved, SNMP is more efficient than Web services.
- In case many objects should be retrieved, Web services may be more efficient than SNMP.
- The encoding and decoding of SNMP PDUs or WSDL messages is less expensive (in terms of processing time) than retrieving management data from within the kernel. The choice between Basic Encoding Rules (BER) or XML encoding is therefore not the main factor that determines performance.
- Performance primarily depends on the quality of the implementation. Good implementations perform considerably better than poor implementations. This holds for SNMP as well as Web services implementations.

In the literature, several studies can be found on the performance of Web services-based management. Choi and Hong published several papers in which they discuss the design of XML-to-SNMP gateways. As part of their research, the performance of XML and SNMP-based management software was investigated [18][19]. To determine bandwidth, they measured the XML traffic on one side of the gateway, and the corresponding SNMP traffic on the other side. In their test setup, separate SNMP messages were generated for every managed object that had to be retrieved; the possibility to request multiple managed objects via a single SNMP message was not part of their test scenario. The authors also measured the latency and resource (CPU and memory) usage of the gateway. They concluded that, for their test setup, XML performed better than SNMP.

Whereas the previous authors compared the performance of SNMP to XML, Neisse and Granville focused on SNMP and Web services [61]. As the previous authors, they implemented a gateway and measured traffic on both sides of this gateway. Two gateway translation schemes were used: protocol level and managed object level. In the first scheme, a direct mapping exists between every SNMP and Web services message. In the second scheme, a single, high-level Web service operation (e.g., `getIfTable`) maps onto multiple SNMP messages. These authors also investigated the performance impact of using HTTPS and `zlib` compression [94]. The Web services toolkit was NuSOAP. For protocol-level translation, they concluded that Web services always require substantial more bandwidth than SNMP. For translation at the managed-object level, they found that Web services perform better than SNMP if many managed objects were retrieved at a time.

Another interesting study on the performance of Web services and SNMP was conducted by Pavlou *et al.* [75][76]. Their study is much broader than a performance study and also includes CORBA-based approaches. They consider the retrieval of Transmission Control Protocol (TCP) MIB variables and measure bandwidth and round-trip time. Unlike in the previous two studies, no gateways are used, but a dedicated Web services agent was implemented using the Systinet WASP toolkit [90]. The performance of this agent was compared to a Net-SNMP agent [62]. The authors neither investigated alternate SNMP

agents, nor the effect of fetching actual management data in the system. They concluded that Web services causes more overhead than SNMP.

In the remainder of this section, we discuss a study performed in 2004 at University of Twente by one of the authors of this chapter; more details can be found in [77]. For this study, measurements were performed on a Pentium 800 MHz PC running Debian Linux. The PC was connected via a 100 Mbit/s Ethernet card. Four Web services prototypes were build: one for retrieving single OIDs of the `ifTable`, one for retrieving `ifTable` rows, one for retrieving `ifTable` columns and one for retrieving the entire `ifTable` (see Section 5.4). To retrieve management data from the system, we decided to use the same code for the SNMP and Web services prototypes. In this way, differences between the measurements would only be caused by differences in SNMP and Web services handling. For this reason, the Web services prototypes incorporated parts of the Net-SNMP open-source package. All SNMP-specific code was removed and replaced by Web services-specific code. This code was generated using the gSOAP toolkit version 2.3.8 [25][26], which turned out to be quite efficient. It generates a dedicated skeleton and does not use generic XML parsers such as the Document Object Model (DOM) or the Simple API for XML (SAX). For compression, we used `zlib` [94]. The performance of the Web services prototypes was compared to more than twenty SNMP implementations, including agents on end-systems (Net-SNMP, Microsoft Windows XP agent, NuDesign and SNMP Research's CIA agent) as well as agents within routers and switches (Cisco, Cabletron, HP, IBM and Nortel).

BANDWIDTH USAGE

One of the main performance parameters is bandwidth usage. For SNMP, it can be derived analytically as the PDU format of SNMP is standardized. For Web services, there were no standards when we conducted this study, so we could not derive analytical formulae that would give lower and upper bounds for the bandwidth needed to retrieve `ifTable` data. In fact, the required bandwidth depends on the specific WSDL description, which may be different from case to case. This section therefore only discusses the bandwidth requirements of the prototype that fetches the entire `ifTable` in a single interaction; the WSDL structure of that prototype was presented in Section 5.4.

Figure 17 shows the bandwidth requirements of SNMPv1 and Web services as a function of the size of the `ifTable`. Bandwidth usage is determined at the application (SNMP/SOAP) layer; at the IP layer, SNMP requires 56 additional octets, whereas Web services require between 273 and 715 octets, depending on the number of retrieved managed objects and the compression mechanism applied. In case of SNMPv3, an additional 40 to 250 octets are needed.

Figure 17 illustrates that SNMP consumes far less bandwidth than Web services, particularly in cases where only a few managed objects should be retrieved. But, even in cases where a large number of objects should be retrieved, SNMP remains two (`get`) to four (`getBulk`) times better than Web services. The situation changes, however, when compression is used. Figure 17 shows that, for Web services, compression reduces bandwidth consumption by a factor 2 when only a single object is retrieved; if 50 objects are retrieved, by a factor 4; for 150 objects, by a factor 8.

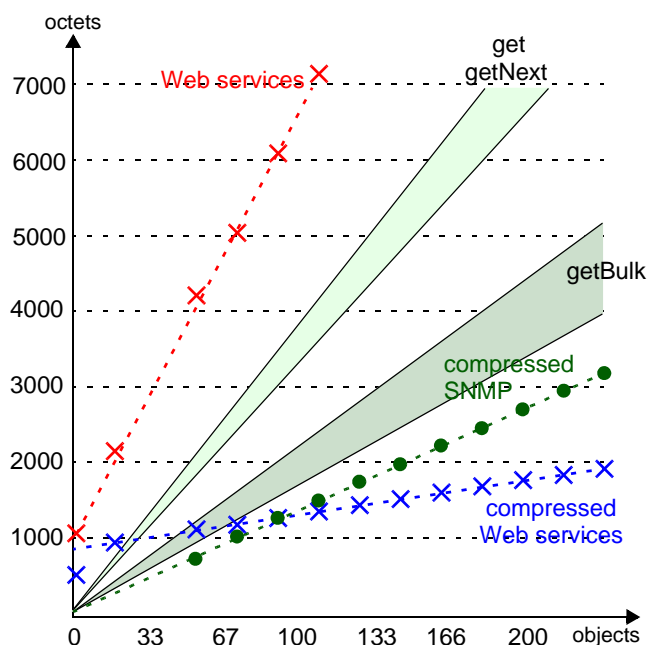


Figure 17: Bandwidth usage of SNMP and Web Services

The threshold where compressed Web services begin to outperform SNMP is 70 managed objects. This number is not ridiculously high: it is easily reached when interface data needs to be retrieved for hundreds of users, for example in case of access switches or Digital Subscriber Line Access Multiplexers (DSLAMs). Figure 17 also shows that the bandwidth consumption of compressed SNMP, as defined by the IETF EoS Working Group, is approximately 75% lower than non-compressed SNMPv1. This form of SNMP compression is known under the name ObjectID Prefix Compression (OPC).

CPU TIME

Figure 18 shows the amount of CPU time needed for coding (decoding plus the subsequent encoding) an SNMP message (encoded with BER) or Web services (encoded in XML) encoded messages. As we can see, the different curves can be approximated by piecewise linear functions. For an SNMP message carrying a single managed object, the BER coding time is roughly 0.06 ms. A similar XML-encoded message requires 0.44 ms, which is seven times more. Coding time increases if the number of managed objects in the message increases. Coding an SNMP message that contains 216 objects requires 0.8 ms; coding a similar Web services message requires 2.5 ms. We can therefore conclude that XML encoding requires 3 to 7 times more CPU time than BER encoding. Figure 18 also shows the effect of compression: compared to XML coding, Web services compression turns out to be quite expensive (by a factor 3 to 5). In cases where bandwidth is cheap and CPU time expensive, Web services messages should not be compressed.

To determine how much coding contributes to the complete message handling time, the time to retrieve the actual data from the system was also measured. For `ifTable`, this

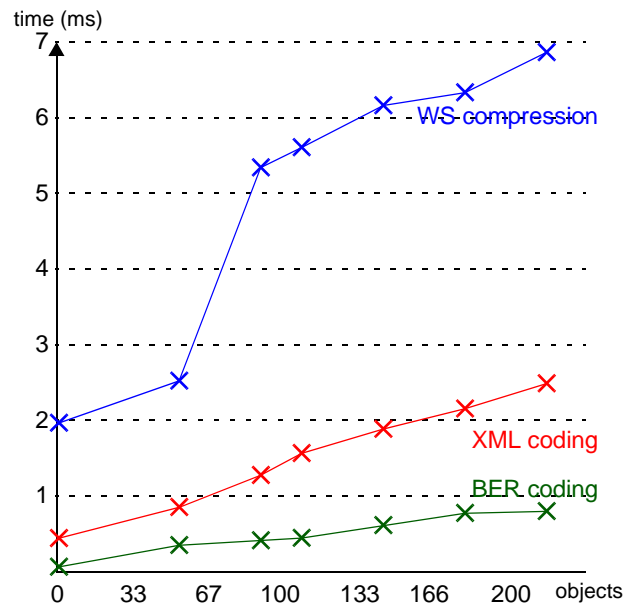


Figure 18: CPU time for coding and compression

retrieval requires, among other things, a system call. We found out that retrieving data is relatively expensive; fetching the value of a single object usually takes between 1.2 and 2.0 ms. This time is considerably larger than the time needed for coding.

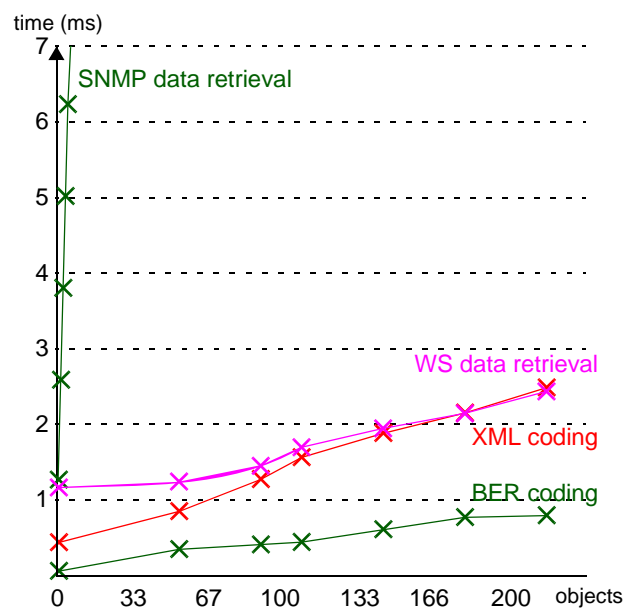


Figure 19: CPU time for coding and data retrieval

The results are depicted in Figure 19; to facilitate the comparison, the BER and XML coding times are mentioned as well. The figure shows that data retrieval times for Net-SNMP increase quite fast; five managed objects already require more than 6 ms; for retrieving 54 managed objects, 65 ms were needed; and retrieving 270 managed objects from the operating system took 500 ms! These staggering times can only be explained by the fact that Net-SNMP performs a new system call every time it finds a new `ifTable` object within the `get` or `getBulk` request; Net-SNMP does not implement caching. The conclusion must therefore be that data retrieval is far more expensive than BER encoding; CPU usage is not determined by the protocol handling, but by the quality of the implementation.

MEMORY USAGE

Memory is needed for holding program instructions (“instruction memory”) and data (“data memory”). For data memory, a distinction needs to be made between static and dynamic memory. Static memory is allocated at the start of the program and remains constant throughout the program lifetime. Dynamic memory is allocated after a request is received and released after the response is transmitted. If multiple requests arrive simultaneously, dynamic memory is allocated multiple times.

Figure 20 shows the memory requirements of Net-SNMP and our Web services prototype. We see that Net-SNMP requires roughly three times as much instruction memory than the Web services prototype. Net-SNMP requires 20 to 40 times more data memory, depending on the number of objects contained in the request. These large numbers, much larger than their Web services counterparts, should not be misinterpreted: the functionality of Net-SNMP is much richer than that of our Web services prototype. For instance, Net-SNMP supports three different protocol versions, includes encryption, authentication and access control, and is written in a platform-independent way.

	instructions	data	
		static	dynamic
SNMP	1972 KB	128 KB	70 - 160 KB
Web services	580 KB	470 B	4 KB

Figure 20: Memory requirements

ROUND-TRIP TIME

In the previous subsections, the performance of the `getIfTable` Web service was compared to that of Net-SNMP. This subsection compares the performance of this Web services prototype to other SNMP implementations. As we could not add code to existing agents code for measuring BER encoding and data retrieval times, we measured instead the

round-trip time, as a function of the number of retrieved managed objects. This time can be measured outside the agent and thus does not require any agent modification.

	1	22	66	270
WS	1,7	2,6	10,3	36,5
WS-Comp	3,3	4,3	5,6	11,8
SNMP-1	1,0	2,5		
SNMP-2	1,0	2,7		
SNMP-3	1,1	3,2	6,9	14,9
SNMP-4	2,0	15,2		
SNMP-5	3,7	18,3		
SNMP-6	1,3	18,9	53,0	
SNMP-7	2,9	58,0	167,0	695,0
SNMP-8	3,1	58,2	168,8	700,0
SNMP-9	3,5	62,0	183,8	767,0
SNMP-10	5,0	80,9		
SNMP-11	12,0	86,9	244,3	
SNMP-12	12,4	257,9		

Figure 21: Round-trip time (in ms)

Figure 21 shows the results for 12 different SNMP agents, as well as the `getIfTable` Web service prototype. It is interesting to note that the round-trip time for normal Web services increases faster than that of compressed Web services. For most SNMP measurements, we used `getBulk` with max-repetition values of 1, 22, 66 and 270; for agents that do not support `getBulk`, a single `get` request was issued, asking for 1, 22, 66 or 270 objects. Not all agents were able to deal with very large size messages; the size of the response message carrying 270 managed objects is such that, on an Ethernet Local Area Network, the message must be fragmented by the IP protocol. As the hardware for the various agents varied, the round-trip times showed in Figure 21 should be used with great care and only considered purely indicative. Under slightly different conditions, these times may be quite different. For example, the addition of a second Ethernet card may have a severe impact on round-trip times. Moreover, these times can change if managed objects other than `ifTable` OIDs are retrieved simultaneously. In addition, the first SNMP interaction often takes considerably more time than subsequent interactions.

Despite these remarks, the overall trend is clear. With most of the SNMP agents that we tested, the round-trip time heavily depends on the number of managed objects that are retrieved. It seems that several SNMP agents would benefit from some form of caching; after more than 15 years of experience in SNMP agent implementation, there is still room for performance improvements. From a latency point of view, the choice between BER and XML encoding does not seem to be of great importance. Our Web services prototype performs reasonably well and, for numerous managed objects, even outperforms several commercial SNMP agents.

6 TOWARD COARSE-GRAINED WEB SERVICES FOR INTEGRATED MANAGEMENT

In the previous section, Web services were used in SNMP-based network management as getters and setters of fine-grained MIB OIDs. Similarly, in a WBEM environment, they could be used as getters and setters of CIM object properties, or they could be mapped to fine-grained CIM operations. This is the granularity at which Web services-enabled management platforms generally operate today. For instance, HP's Web Service Management Framework [14] specifies how to manipulate fine-grained managed objects with Web services. But is this the way Web services are meant to be used as management application middleware?

Web services did not come out of the blue, context free: they were devised with SOA in mind. SOA is an IT architectural style, to use Fielding's terminology [28], and Web services implement it. In this section, we go back to the rationale behind SOA, consider management applications from this perspective, and show that Web services could be used differently in integrated management. We illustrate this claim with examples of coarse-grained Web services for integrated management.

6.1 SERVICE-ORIENTED ARCHITECTURE

Defining SOA poses two problems. First, there is no single and widely shared definition. No one clearly defined this concept before it began pervading the software industry. This concept grew and spread by hearsay in the software engineering community, as a conceptualization of what Web services were trying to achieve. The definitions proposed by the W3C and OASIS came late, and despite the credit of these organizations in the community, none of their definitions are widely accepted.

Second, since 2004–2005, SOA has turned into a buzzword in the hands of marketing people. Today, there are almost as many definitions of this acronym as vendors in the software industry. There are also countless examples of software tools and packages that have been re-branded "SOA-enabled" without supporting all the features that engineers generally associate with this acronym.

In an attempt to distinguish between a stable technical meaning and ever-changing marketing blurb, some authors use the term Service-Oriented Computing (SOC) [71][80] to refer, among engineers, to the distributed computing paradigm, leaving SOA to hype. Other authors consider that SOC is a distributed computing paradigm whereas SOA is a software architecture; the subtle difference between the two is not always clear. In this chapter, we consider these two acronyms synonymous and use SOA in its technical sense.

So, what is SOA about? Because no one "owns" its definition, we reviewed the literature and browsed a number of Web sites to form an opinion.

In WSA [9] (defined in Section 2.5), the W3C defines SOA as "a form of distributed systems architecture" that is "typically characterized" (note the implied absence of consensus) by the following properties:

- *logical view*: the service is an abstract view of actual programs, databases, business processes, etc.;
- *message orientation*: the service is defined in terms of messages exchanged between consumers and providers;
- *description orientation*: the service is described by machine-parsable metadata to facilitate its automated discovery;
- *granularity*: services generally use a small number of operations with relatively large and complex messages;
- *network orientation*: services are often used over a network, but this is not mandatory;
- *platform neutral*: messages are sent in a standard format and delivered through interfaces that are independent of any platform.

In August 2006, OASIS released its Reference Model for SOA (SOA-RM) [50]. This specification was long awaited by the industry. Unfortunately, this version 1.0 is quite verbose and rather unclear, and often proposes overly abstract definitions; we hope that OASIS will solve these teething problems in subsequent versions. Difficult to read as it is, this document still clarifies three important points:

- “SOA is a paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains”;
- a service is accessed via an interface;
- a service is opaque: its implementation is hidden from the service consumer.

Channabasavaiah *et al.* [15] are more practical and define SOA as “an application architecture within which all functions are defined as independent services with well-defined invocable interfaces, which can be called in defined sequences to form business processes”. They highlight three aspects:

- all functions are defined as services;
- all services are independent and operate as black boxes;
- consumers just see interfaces; they ignore whether the services are local or remote, what interconnection scheme or protocol is used, etc.

IBM’s online introduction to SOA [37] lists the following characteristics:

- SOA is an IT architectural style;
- integrated services can be local or geographically distant;
- services can assemble themselves on demand; they “coalesce to accomplish a specific business task, enabling (an organization) to adapt to changing conditions and requirements, in some cases even without human intervention”; coalescence refers to automated service composition at run-time;
- services are self-contained (i.e. opaque) and have well-defined interfaces to let consumers know how to interact with them;
- with SOA, the macro-components of an application are loosely coupled; this loose coupling enables the combination of services into diverse applications;
- all interactions between service consumers and providers are carried out based on service contracts.

Erl [27] identifies four core properties and four optional properties for SOA:

- services are autonomous (core);
- loose coupling (core);
- abstraction: the underlying logic is invisible to service consumers (core);
- services share a formal contract (core);
- services are reusable;
- services are composable;
- services are stateless;
- services are discoverable.

O'Brien *et al.* [66] complement this list of properties with the following two:

- *Services have a network-addressable interface*: “Service (consumers) must be able to invoke a service across the network. When a service (consumer) and service provider are on the same machine, it may be possible to access the service through a local interface and not through the network. However, the service must also support remote requests.”
- *Services are location transparent*: “Service (consumers) do not have to access a service using its absolute network address. (They) dynamically discover the location of a service looking up a registry. This feature allows services to move from one location to another without affecting the (consumers).”

Singh and Huhns [80] characterize SOA as follows:

- *loose coupling*;
- *implementation neutrality*: what matters is the interface; implementation details are hidden; in particular, services do not depend on programming languages;
- *flexible configurability*: services are bound to one another late in the process;
- *long lifetime*: interacting entities must be able to cope with long-lived associations; services should live “long enough to be discovered, to be relied upon, and to engender trust in their behavior”;
- *coarse granularity*: actions and interactions between participants should not be modeled at a detailed level;
- *teams*: participants should be autonomous; instead of a “participant commanding its partners”, distributed computing in open systems should be “a matter of business partners working as a team”.

For these authors, the main benefits of SOA are fourfold [80]:

- services “provide higher-level abstractions for organizing applications in large-scale, open environments”;
- these abstractions are standardized;
- standards “make it possible to develop general-purpose tools to manage the entire system lifecycle, including design, development, debugging, monitoring”;
- standards feed other standards.

All these definitions (and many others not mentioned here) draw a fairly consistent picture. SOA is a software paradigm, an IT architectural style for distributing computing with the following properties:

- *Loose coupling*: In SOA, distributed applications make use of loosely coupled services.
- *Coarse-grained*: Services have a coarse granularity, they fulfill some functionality on their own.
- *Autonomous*: Services are stateless and can be used by many applications or other services.
- *Interfaces and contracts*: Services are opaque entities accessed via interfaces on the basis of a high-level contract; implementation details are hidden.
- *Network transparency*: Services can be local or remotely accessed via the network. The location of the service provide may change over time.
- *Multiple owners*: Services can be owned by different organizations.
- *Service discovery*: Services are discovered dynamically, possibly automatically, at run-time; service providers are not hard-coded in applications.
- *Service composition*: Services can be composed transparently, possibly dynamically at run-time.

There is also a large consensus about three things. First, Web services are just one technology implementing SOA. Other technologies may come in the future that could implement SOA differently, perhaps more efficiently. For instance, one could imagine that SOA be implemented by a new variant of CORBA that would (i) define services as high-level CORBA objects wrapping up CORBA and non-CORBA legacy applications, (ii) use the Internet Inter-ORB Protocol (IIOP) instead of SOAP and HTTP to make service consumers and providers communicate, (iii) advertise services in the form of interfaces (expressed in CORBA IDL) in standard external registries (as opposed to CORBA-specific registries), (iv) enable these services to be discovered and composed dynamically at run-time using semantic metadata, and (v) solve the poor interoperability and the performance problems of multi-vendor inter-ORB communication.

Second, nothing in SOA is new. For instance, interfaces that make it possible to access opaque entities were already defined in Open Distributed Processing [39]. What is new is that we now have the technology to make it work: “SOAs are garnering interest, not because they are a novel approach, but because Web services are becoming sufficiently mature for developers to realize and reevaluate their potential. The evolution of Web services is at a point where designers can start to see how to implement true SOAs. They can abstract a service enough to enable its dynamic and automatic selection, and Web services are finally offering a technology that is rich and flexible enough to make SOAs a reality” [49].

Third, a key difference between Web services and distributed object middleware is that Web services are not objects that need other objects for doing something useful: they are autonomous applications fulfilling some functionality on their own. As a result, Web services should not be regarded as yet another type of object-oriented middleware competing with CORBA or J2EE [89]: they are supposed to operate at a higher level of abstraction.

It should be noted that this coarse-grained approach to middleware poses problems to the software industry, because hordes of engineers are used to thinking in terms of tightly coupled objects, and a plethora of tools work at the object level [89]. Even some academics struggle to adjust their mindframe [8].

6.2 COMPARISON BETWEEN SOA AND CURRENT PRACTICE

Now that we better understand the SOA approach to distributed computing, let us compare, one by one, the principles that underpin Web services with the current practice in integrated management. By “practice”, we mean the management platforms that enterprises actually use to manage their networks, systems and services, as opposed to the latest commercial offerings from vendors who may claim to support SOA.

APPLICATIONS MAKE USE OF SERVICES

SOA and integrated management both have the concept of *service*; they give it slightly different but reasonably consistent meanings. For both, we have low-level services that abstract certain parts of the IT infrastructure (e.g., a network service for an MPLS-based VPN) and high-level services that are user-centered and achieve a specific business goal (e.g., a business service for online book ordering).

A Web service does not necessarily pertain to Service-Level Management (SLM): it may be a building block for fulfilling network management tasks, or systems management tasks. In this case, the SOA concept of service corresponds roughly to the concept of management task in integrated management.

In integrated management, a high-level service embodies a contract between an end-user (a person) and a service provider (a vendor). In TMN, a network service is captured by the Service Management Layer (SML), while the contract between a user and a vendor belongs to the Business Management Layer [35]. The SML is where network and business services are tied together. In SOA, the service contract binds a service consumer to a service producer.

Both in SOA and integrated management, services can be characterized by a Service-Level Agreement (SLA), which captures the end-users’ perception of whether the IT infrastructure works well for achieving a specific business goal. High-level services serve as intermediaries between business views and IT views; they enable end-users to abstract away the infrastructure and consider it as a black box.

LOOSE COUPLING

Integrated management platforms use both tight and loose forms of integration. We can distinguish five cases.

First, communication between managed entities (agents) and managing entities (managers) is loosely coupled. It relies on standard management protocols (e.g., SNMP) that hide implementation details by manipulating abstractions of real-life resources known as managed objects.

Second, the *macro-components* of a monitoring application (event correlator, event filter, event aggregator, data analyzer, data filter, data aggregator, data poller, topology discovery engine, network map manager, etc. [53]) are tightly integrated. They are provided by a single vendor. Communication between these macro-components is proprietary, using all sorts of programming tricks to boost performance (e.g., the maximum number of events that can be processed per time unit).

Third, when management is distributed over multiple domains (e.g., for scalability reasons in large enterprises, or budget reasons in corporations with financially independent subsidiaries), manager-to-manager communication is proprietary and relies on tight coupling in the SNMP world; in the WBEM world, it is usually loosely coupled and based on standard management protocols. Note that exchanging management information with external third parties requires sophisticated firewalls in the case of tight coupling, and simpler firewalls in the case of loose coupling.

Fourth, communication between the monitoring application and the data repository is sometimes loosely coupled, sometimes tightly coupled. The data repository is logically unique but can be physically distributed; it consists typically of a relational database for monitoring data and a Lightweight Directory Access Protocol (LDAP) repository for configuration data; sometimes it may also include an object base, or an XML repository, or a knowledge base, etc. An example of loose coupling¹ is when the monitoring application accesses a relational database via a standard interface such as Open Data Base Connectivity (ODBC) using a standard query language such as the Structured Query Language (SQL); the actual data repository used underneath (Oracle database, Sybase database, etc.) is then transparent to the application. An example of tight coupling is when the application uses a proprietary query language (e.g., a so-called “enhanced” version of SQL, with proprietary extensions to the language). Vendor lock-in always implies tight coupling.

Fifth, in large enterprises or corporations, people not only use monitoring applications, but also data mining applications that perform trend analysis, retrospective pattern analysis, etc. Examples include Enterprise Resource Planning (ERP) to anticipate the saturation of resources, security analysis to search for attack patterns, event analysis to identify the important cases that need to be fed to a case-based reasoning engine, etc. Communication between the data repository and these data mining applications is usually loosely coupled: people use standard interfaces and protocols for exchanging data because these applications are generally provided by third-party vendors. For performance reasons, data mining applications may occasionally be tightly coupled to the data repository.

SERVICES ARE COARSE-GRAINED

How do the SOA concepts of “fine-grained services” and “coarse-grained services” translate into integrated management? In the SOA approach to distributed computing, we saw that applications make use of services, which isolate independent pieces of the application that can be reused elsewhere (e.g., by other applications or other services). Some of these services are conceptually at a high level of abstraction (e.g., “Balance the load of virtual server VS among the physical servers PS1, PS2, PS3 and PS4”), others at a low level (e.g., “Set the administrative status of CPU C to down”). Functionality is thus split between the application and reusable services of varying levels of abstraction.

In integrated management, this corresponds to management tasks in the functional model [53]. Functionality is split into management tasks, some of which are low level, some of which are high level. Some of these functions are generic (e.g., see the standard TMN

1. Note that we use “loose coupling” in a flexible manner here, compared to SOA: by using SQL, the application cannot interact with a generic data repository, but only with a relational database.

management functions [41]) and can in principle be separated from the application. The common practice in telecommunications is that management applications are based on CORBA and implement a mixture of OSI Management, TMN and SNMP. Generic functions are accessed via CORBA interfaces as independent low-level services, which can be considered coarse-grained because they do not simply operate at the managed object level. In enterprises, however, the situation is very different: today's management platforms are based on SNMP or WBEM (which do not define standard generic functions), not on OSI Management or TMN; they are typically implemented in C, C++, Java or C#, and based neither on CORBA nor on J2EE. As a result, they do not handle separate services.

In short, in the enterprise world, management tasks are not implemented in the form of independent coarse-grained services. This is a major difference with SOA.

From outside, management applications appear to operate at only two scales: the managed object level (when interacting with managed entities), and the management application level (the monitoring application globally supports some functionality as a big black box). There are no intermediaries that could easily be mapped to Web services.

Internally, management applications actually operate at more scales. When they are coded in an object-oriented programming language, these scales are: managed object, object, software component, thread, process, and complete application. With procedural programming languages, these scales are: managed object, function, thread, process, and complete application.

SERVICES ARE AUTONOMOUS

We saw in Section 1.2 that large enterprises generally run several management platforms in parallel. These applications are autonomous in the SOA sense, but they are not packaged as services. They usually do not communicate directly (there are exceptions), but they all (at least supposedly) send meaningful events to the same event correlator, which is in charge of integrating management for the entire enterprise. In some cases, this logically unique event correlator can be physically distributed [55].

Network management platforms (e.g., HP Openview and IBM Tivoli) often accept equipment-specific graphical front-ends (so-called "plug-ins") provided by third-party vendors. These extensions can be seen as autonomous services that are statically discovered and integrated.

INTERFACES AND CONTRACTS

Integrated management applications work with two types of interface: interfaces to their internal components, and interfaces to managed entities (managed objects). The former do not hide implementation details (on the contrary, these components can expect other components to support programming language-specific features such as Java exceptions, for instance), but the latter do (managed objects are standard and hide the idiosyncrasies of the underlying managed resource).

In SOA, services accessed via interfaces form the basis for a contract. In integrated management, contracts are captured in the form of SLAs.

NETWORK TRANSPARENCY

In SOA, network transparency means that services can be local or remote. This transparency has no equivalent in management platforms. For obvious confidentiality and robustness reasons, no one wants a monitoring application to discover automatically, and later use, an event correlator provided by an unknown organization on the opposite side of the planet!

MULTIPLE OWNERS

In integrated management, the monitoring and data mining applications are generally run by the same organization, in which case there is only one owner. In large organizations, these applications can be distributed over a hierarchy of domains; different management applications are then run by different teams, possibly attached to different profit centers; but they all really belong to the same organization, the same owner from an SOA standpoint.

When organizations outsource partially or totally the management of their IT infrastructure and/or the management of their services, there are clearly multiple owners. Relationships between them are controlled by business contracts, and not just Web services contracts.

Extranets and subsidiaries are typical examples where we have different owners. In an extranet, a corporation works closely with a large set of subcontractors and/or retailers. This requires the IT infrastructures of different owners to be tightly coupled. The situation is similar in corporations that have financially independent subsidiaries. Although the latter are different enterprises, different owners, they have strong incentives for cooperating as if they were a single organization, a single owner in SOA parlance.

SERVICE DISCOVERY

In today's integrated management applications, only very simple forms of service discovery are generally implemented, primarily for retrieving configuration settings. In systems management, LDAP repositories are commonly used.

SERVICE COMPOSITION

Integrated management platforms generally found in businesses today do not use dynamic service composition at run-time. Management tasks are composed statically in the monitoring application or the data mining applications, and cast in iron in static code.

6.3 ANALYSIS

The first outcome of our comparison is that the management platforms deployed in real life are far from implementing SOA natively. Instead of simply using Web services to exchange managed objects or events, as suggested by recent standardization activities such as WSDM or WS-Management, we should rethink (i.e., re-architect and re-design) management applications to make them SOA compliant. This is a classic mission for software architects.

The main differences between SOA and today's practice in integrated management are (i) the absence of autonomous coarse-grained management tasks in management platforms, (ii) the tight coupling of the monitoring application (most if not all of its macro-components communicate via proprietary mechanisms), and (iii) the quasi absence of service discovery (except for retrieving configuration settings).

A fourth difference, the lack of dynamic service composition at run-time, is less of a problem because, strictly speaking, service composition is not a requirement in SOA: it is simply desirable, and the odds are that when the first three problems are solved, people will find ways of composing these new services.

Reconciling the first three differences identified above requires modifying not only the way management applications are structured, but also the management architectures that underpin them (notably SNMP and WBEM).

CHANGES IN THE MANAGEMENT ARCHITECTURE

In integrated management, it is customary to decompose a *management architecture* (also known as *management framework*) into four models [30]: (i) an organizational model, which defines the different entities involved in a management application and their roles; (ii) an information model, which specifies how resources are modeled in the management domain; (iii) a communication model, which defines how to transfer data (e.g., a communication pipe and service primitives for exchanging data across this pipe); and (iv) a functional model, which defines functions (management tasks) that do real work; these functions can be fine-grained (e.g., a generic class for managing time series) or coarse-grained (e.g., an event correlator). Migrating to SOA requires changes in all but one of these models.

First, the *communication model* needs to migrate to SOAP. This migration would be straightforward for WBEM, whose communication model is close to SOAP. For SNMP, as we saw in Section 5, it would imply the adoption of another standard communication pipe (typically HTTP, possibly another one) and new communication primitives (SOAP RPCs).

Second, SOA challenges the *organizational model* of the management architectures used to date: SNMP, WBEM, TMN, OSI Management, etc. The management roles (defined in the manager-agent paradigm [74]) need to be revisited to make integrated management comply with SOA. First and foremost, the concept of coarse-grained autonomous service needs to be introduced. Migrating to SOA is also an invitation to stop interpreting the manager-agent paradigm as meaning "the manager should do everything and the agent nothing", as already advocated by Management by Delegation [92] in the 1990s but never quite enforced.

As far as the *functional model* is concerned, SOA encourages a new breakdown of management tasks among all actors involved. Instead of bringing all monitoring data and all events to the monitoring application and doing all the processing there, in a large monolithic application, SOA suggests to define intermediary tasks and wrap them as independent and reusable services that can run on different machines. This too is somewhat reminiscent of Management by Delegation [92], the Script MIB [46] in SNMP or the Command Sequencer [40] in OSI Management, except that Web services should be full-fledged applications, not just small scripts with a few lines of code.

CHANGES IN THE MANAGEMENT APPLICATIONS

Assuming that the main management architectures used to date (SNMP and WBEM) are updated, what should be changed in management applications to make them implement autonomous coarse-grained services, loosely coupled services and service discovery?

The first issue (autonomous coarse-grained services) is about delegation granularity in a management application, i.e. how this application is subdivided into management tasks and how these tasks are allotted to different entities for execution. A few years ago, one of the authors of this chapter used organization theory to identify four criteria for classifying management architectures: semantic richness of the information model, degree of granularity, degree of automation of management, and degree of specification of a task [53]. Based on that, we defined an “enhanced” taxonomy of network and systems management paradigms. For the delegation granularity, four scales were proposed in our taxonomy: no delegation, delegation by domain, delegation by microtask and delegation by macrotask. Fine-grained Web services correspond to microtasks; coarse-grained Web services correspond to macrotasks. To be SOA compliant, management applications should therefore support a new form of delegation: macrotasks.

More specifically, we propose that management applications use Web services at three scales, three levels of granularity. First, *simple management tasks* are slightly coarser-grained than Web services operating at the managed-object level. Instead of merely getting or setting an OID, they can execute small programs on a remote managed entity. Their execution is instantaneous. The Web service providers have a low footprint on the machine where they run. Second, *elaborate management tasks* operate at a higher level of granularity and consist of programs that do something significant. Their execution is not instantaneous and may require important CPU and memory resources. Third, *macro-components* (see examples in Section 6.2, “Loose Coupling”) communicate via Web services instead of proprietary mechanisms. This last change makes it easy to distribute them across several machines, which complements domain-based distributed hierarchical management.

Once we have autonomous coarse-grained services, the second issue (loosely coupled services) is easy to deal with: management applications just need to use Web services and SOAP for communicating with these services. This guarantees a loose coupling of the management tasks.

To address the third issue (service discovery), we propose to let managing and managed entities discover one another at run-time, using Web services. This enables managed entities to be associated dynamically with a domain, instead of being configured once and for all. This change is also in line with recent standardization activities (e.g., WSDM).

6.4 HOW TO MAKE MANAGEMENT PLATFORMS SOA COMPLIANT

Let us now go through a series of examples that illustrate how an SOA-enabled management platform could work.

TYPE 1: SIMPLE MANAGEMENT TASKS AS WEB SERVICES

Two examples of simple management tasks are the invocation of non-trivial actions and the execution of simple scripts.

Example 1: Remote execution of a non-trivial action

Non-trivial actions can be seen as RPCs bundled with a bit of code. The reboot of a remote device is a good example of action that can be invoked by a management application as a Web service delegated to a third party (neither the manager, nor the agent in SNMP parlance). In this Web service, we first try to do a clean (“graceful”) reboot by setting an OID or invoking an operation on a CIM object. If it does not work, we then trigger a power-cycle of the machine. This operation can translate into a number of finer-grained operations, depending on the target machine to be rebooted. For instance, to power-cycle a remote PC that is not responding to any query anymore, we can look for an electromagnet that controls its power. If we find one (e.g., in a configuration database or a registry), we then set an OID or invoke a CIM operation on this electromagnet, depending on the management architecture that it supports, to force the power-cycle. A few moments later, the Web service checks whether the PC has resumed its activities; it sends back a response to the management application, with status succeeded or failed.

Using a Web service for this task allows the management application to abstract away this complexity, which is equipment specific. Binding a given piece of equipment to the right reboot Web service then becomes a matter of configuration or publication in a registry.

Example 2: Simple script

An example of simple script is the retrieval of routing tables from a group of routers. If multiple routing protocols are supported by a router, all of its routing tables need to be downloaded. This information can be subsequently exploited by the management application for debugging transient routing instabilities in the network. Alternatively, this Web service can also be used on a regular basis for performing routing-based event masking, for instance in networks where routes change frequently (“route flapping”) and network topology-based event masking does not work satisfactorily.

In this example, the Web service providers can be the routers themselves; another possibility is to have a single Web service provider that runs on a third-party machine (e.g., an SNMP gateway or proxy) and interacts via SNMP with the different routers concerned by this task [72].

TYPE 2: ELABORATE MANAGEMENT TASKS AS WEB SERVICES

We now go one level up in abstraction and present four examples of elaborate management tasks: core problem investigation, ad hoc management, policy enforcement check and long-lasting problem investigation.

Example 1: Core problem investigation

In a management platform, when the event correlator has completed the root-cause analysis phase, we are left with a small set of core problems that need to be solved as soon as possible. When the solution to the problem is not obvious, we then enter a phase known as

problem investigation. Different technologies can be used during this phase: an Event-Condition-Action (ECA) rule-based engine, a case-based reasoning engine, etc. On this occasion, the management application can launch programs¹ to retrieve more management information to further investigate, and hopefully solve, each outstanding problem.

Assuming that we do not have performance constraints that impose to keep all processing local², these programs can be delegated to other machines in the form of Web services, thereby contributing to balance the load of the management application across multiple machines and to improve scalability. They run autonomously, gather data independently, access the data repository on their own, etc.

For instance, when the response time of a Web server is unacceptably long for an undetermined reason, it can be useful to monitor in great detail the traffic between this server and all the hosts interacting with it. This can be achieved by a Web service monitoring, say, for 30 minutes all inbound and outbound traffic for this server and analyzing this traffic in detail afterward, using fast data mining techniques. This task can involve many fine-grained operations; for instance, it can retrieve Remote MONitoring (RMON) MIB [85] OIDs in bulk at short time intervals, or it can put a network interface in promiscuous mode and sniff all traffic on the network. The duration of this debugging activity can be a parameter of the Web service operation. Upon successful completion, the Web service returns the cause of the problem; otherwise, it may be able to indicate whether it found something suspicious.

Example 2: Ad hoc management

Ad hoc management tasks [53] are not run regularly by the monitoring application. Instead, they are executed on an *ad hoc* basis to investigate problems as they show up. These tasks can be triggered interactively (e.g., by staff in a NOC), or automatically by the management application (see example 1 with the slow Web server). These tasks are good candidates for Web services: it makes sense to delegate these self-contained and independent programs to third-party machines that do not have the same constraints, in terms of resource availability, as the main management platform that runs the bulk of the management application³.

For instance, if the administrator suspects that the enterprise is currently undergoing a serious attack, he/she may launch a security analysis program that performs pattern matching on log files, analyzes network traffic, etc. This program will run until it is interrupted, and report the attacks that it detects (or strongly suspects) to the administrator using SMS messaging, paging, e-mail, etc.

Example 3: Policy enforcement check

Another interesting case of elaborate management task is to check that policies used in policy-based management [81] are enforced durably. For instance, let us consider a classic source of problems for network operators: `telnet`, the main enemy of policies. The bottom line of this scenario is invariably the same, with variations in the details.

-
1. These programs can be executable binaries, byte-code for virtual machines, scripts, etc.
 2. For instance, that would be the case if we had to interact with thousands of finite state machines only accessible from the address space of the event correlator.
 3. We do not assume a centralized management platform here. When management is distributed hierarchically over multiple managers, the problem is the same as centralized management: each manager can be a bottleneck for its own sub-domain.

A technician in a NOC is in charge of debugging a serious problem. Customers are complaining, the helpdesk is overwhelmed, his boss called him several times and is now standing next to him, watching over his shoulder, grumbling, doing nothing but stressing our technician. The latter feels he is alone on earth. All the management tools at his disposal indicate that there is an avalanche of problems, but they all fail to find the cause of these problems. After checking hundreds of configuration settings on many devices, our technician decides to stop services and reconfigure network cards one by one, hoping that this will eventually point to the problem. Using `telnet`, he modifies the configuration of many operational routers, trying all sorts of things... Doing so, he gradually starts to build a picture of what is going on: he had a similar problem several years ago, a misconfigured router was reinjecting routes when it should not. Today's problem is in fact very different, but our technician is now convinced he knows what is going on. Against all odds, after an hour of frantic typing, the situation starts to improve. Everything is still slow, but there seems to be a way out of this serious problem. And when he modifies the 27th router, bingo, the problem is gone and everything comes back to normal! His boss applauds, thanks the Lord for sending him such a great man, goes back to his office and phones his own boss, to make sure he does not get the blame for this major outage...

Is the incident closed? Yes, because things work again. Is the problem solved? No, because our technician still does not know what the problem was. Is the situation back to normal? Not at all: first, the same problem could resurface at any time; second, by changing the configuration of dozens of devices, our technician has placed several timed bombs in the network, without his knowing it. These timed bombs are big potential problems that are just waiting for an occasion to go off. And why so? Because he cannot roll back all the unnecessary changes that he made. Under stress, he did many things that he cannot remember anymore; and because he is dealing with operational networks, he is now very cautious about modifying anything. Yet he did several changes that he may soon regret. He changed configuration settings that are supposed to be entirely controlled by PDPs, and that will disappear in a few hours because they are overwritten everyday, at the same time, during slack hours. Will the problem resume immediately after the next policy update? He also changed many configuration settings that have nothing to do with the actual problem. Will they cause new problems in the future?

All network operators, all corporations, and presumably most large enterprises have been through this one day. This problem is as old as `telnet`. Under stress, NOC staff can modify (and break) just about everything, irrespective of their skills. One solution is to regularly copy the configuration settings (notably those that enforce policies and are generated automatically) of all network devices in a data store, at least once a day, and analyze every day the changes that were made in the past 24 hours. This process can be partially automated; it is a self-contained and coarse-grained activity, independent of the rest of the management application; it does not have to run on the main management platform. In other words, it should lend itself very well to being ported to Web services. In fact, not only some, but all configuration settings that are under the control of a PDP should systematically be checked like this, by Web services or other means.

In this case, the Web service providers should probably be third-party machines, not operational routers, to avoid temporarily saturating them with management tasks.

Example 4: Long-lasting problem investigation

If we go back to example 1, some of the outstanding core problems may not be solved within a predefined time (e.g., 10 minutes). In such cases, it is appropriate to systematically outsource the resolution of these problems to third-party machines, which take over these investigations and run them as background tasks, possibly taking hours or days to come to a conclusion. Transferring these problem investigations to external entities can be achieved in a loosely coupled and portable manner using Web services. The objective here is to conserve precious few resources on the machine running the bulk of the management application, while not giving up on important but hard problems.

TYPE 3: MACRO-COMPONENTS AS WEB SERVICES

This time, we break up the management application into very large chunks that can be physically distributed over multiple machines. The idea is to implement the network transparency recommended by SOA to improve the scalability of the management application.

Example 1: Distributed monitoring application

In today's management platforms, the main building block of the management application is the monitoring application. We propose to break up this large, monolithic, single-vendor monitoring application into large chunks (macro-components) wrapped up with coarse-grained Web services. These macro-components can be provided by several vendors and integrated on a per-organization basis, depending on site-specific needs and customer-specific price conditions.

This integration seldom requires dynamic binding at run-time, because presumably the macro-components that make up management software will not change often. This restructuring of the monitoring application brings more competition among vendors and can drive costs down (feature-rich management applications capable of integrating the management of networks, systems and services are still incredibly expensive).

For instance, one vendor can provide elaborate graphical views of the network with 3D navigation, while another provides an event correlator with a smart rule description language, and a third provides the rest of the monitoring application. Standard Web service-based communication would enable these loosely-coupled services to work together.

Example 2: Integration of management

Web services can also be useful to integrate the different management platforms in charge of network management, systems management, service management, the trouble-ticket system, technology-specific platforms, etc. Doing so, they have the potential to nicely complement today's management solutions, for instance when the management platforms used in an enterprise cannot use the same event correlator due to integration problems (a serious problem that prevents management from being integrated).

TYPE 4: WEB SERVICES BETWEEN MANAGING/MANAGED ENTITIES

The fourth type of coarse-grained Web services is different from the previous three. Instead of focusing on the delegation granularity, it addresses the issue of service discovery. We present three examples: dynamic discovery of the manager, transparent redundancy of the managing entity, and dynamic discovery of the top-level manager.

Example 1: Dynamic discovery of the manager

In this example, instead of configuring the address of the managing entity in each agent of its domain, administrators rely on Web services for managed entities to discover their managing entity dynamically at run-time. This discovery typically uses a standard registry, either configured in the managed entity or discovered at boot time (e.g., with a multicast operation).

Example 2: Transparent redundancy during discovery

When managed entities think they bind directly to their managing entity, they can in fact connect to an intermediary Web service in charge of improving the robustness of management. This intermediary can transparently support two redundant managers with a hot stand-by (i.e., two machines that receive and analyze the same management data in parallel). The primary is active, the secondary is passive. If the primary fails, the secondary can take over transparently at very short notice. Alternatively, this intermediary can balance the load between multiple managers (which are all active). This can be useful in environments where managers experience important bursts of activity.

Example 3: Dynamic discovery of the top-level manager

In distributed hierarchical management (a technique commonly adopted in large enterprises and corporations [53]), the management domain is under the responsibility of a managing entity known as the *top-level manager*. The domain is logically split into sub-domains, usually for scalability reasons, and each sub-domain is under the responsibility of a managing entity called the *sub-level manager*.

In this scenario, instead of configuring the address of the top-level manager in each sub-level manager, administrators rely on Web services for the sub-level managers to discover automatically the top-level manager of their domain. This operation typically uses a registry.

6.5 MIGRATING TO SOA: IS IT WORTH THE EFFORT?

After studying the changes that the migration to SOA requires in management applications, let us ponder over these changes. Are they reasonable? Do the advantages of SOA outweigh the drawbacks of reengineering existing management applications?

WE NEED SOMETHING NEW

In the last decade, we have seen a number of proposals that challenged the organizational models of existing management architectures: Management by Delegation [92], mobile code [3], intelligent agents [42], multi-agent systems [36], and more recently P2P [86] and Web services. So far, none of them have succeeded in durably influencing the architecture and

design of the management applications used in real life. Why should Web services succeed where other technologies failed?

As we saw in Section 1.2, the main problem with today's management platforms is that we are prisoners of habits that date back to an era when networks were small and simple to manage, resources were scarce, dependencies were not highly dynamic, event correlation was simple because we did not bother to integrate the management of networks, systems and services, etc.

Today, architecting an integrated management application for a corporation is very difficult, and the tools available on the market often fall short of customers' expectations. Designing these complex applications boils down to designing large and highly dynamic distributed applications. This requires the state of the art in software engineering, and the reengineering of integration management applications. Migrating to SOA can be the occasion for this clean slate approach.

Another important reason for adopting coarse-grained Web services is that they offer a solution for integrating multi-vendor dedicated management platforms (e.g., a platform dedicated to network management with another platform specialized in service management). This integration is a weak point of many of today's management platforms, and a number of corporations or large enterprises currently struggle to achieve this integration.

NO GUARANTEE OF SUCCESS

Businesswise, migrating integrated management platforms to SOA is not a guaranteed win. There is much uncertainty. The market of management applications needs to change its habits, and whether this will happen is difficult to predict. Decision makers in the IT industry are told every year that a new "revolution" has just occurred, so they tend to be skeptical. Convincing them will not be easy.

Oddly enough, the main non-technical driver for change may be the comfort offered by homogeneity: if everyone in the software industry adopts SOA, then there is a strong incentive for management applications to also migrate to SOA: few people enjoy being the last dinosaur in town...

PARADOX

Migrating management platforms to Web services leads to an apparent paradox. On the one hand, SOA is about preserving investments in legacy systems by wrapping up existing applications with Web services. The goal here is to avoid the high cost of reengineering applications, especially complex ones. Based on this principle, using Web services in management applications should not require profound changes in these applications: we should not have to reengineer them. On the other hand, using Web services as getters and setters of managed objects defeats the point of coarse-grained and autonomous services in SOA. What is the catch?

This paradox is only apparent. It is due to a confusion between applications and services in SOA parlance. Services are meant to be reusable by several applications or other services; applications themselves are not. Management applications are not used as services by other

applications: they are at the top of the food chain, so to say. Once SOA-enabled, applications can make use of different Web services without being entirely reengineered. But there is no free lunch in IT: they need to be made SOA-enabled in the first place.

PERFORMANCE, LATENCY

The performance of Web service-based management solutions is still very much an open issue. Web services use SOAP messages underneath. This messaging technology, based on the exchange of XML documents, is notoriously inefficient [44]. In corporations and large organizations, integrated management platforms have drastic requirements in terms of performance. Breaking the management application into Web services-enabled macro-components, elaborate management tasks and simple management tasks may turn out to be totally impractical for performance reasons.

The question is particularly acute for communication between macro-components. For instance, wrapping up the event filter, the event aggregator and the event correlator with different Web services could lead to a sharp decrease in performance, because these macro-components need to exchange many events per time unit in large environments, latency must be kept low, and SOAP message exchanges are slow compared to the current tightly coupled practice. A performance analysis of management platforms using coarse-grained Web services is outside the scope of this chapter, but this matter deserves attention in follow-up work.

7 CONCLUSION

In this chapter, we summarized the main standards pertaining to Web services and presented different ways of using this middleware technology in integrated management. Fine-grained Web services are a direct mapping of current habits in management: each Web service operates at the managed object level. This is the approach currently adopted by standards bodies (e.g., in WSDM and WS-Management). We showed that it is efficient when many compressed managed objects are transferred in bulk. Unfortunately, it does not comply with SOA, the distributed computing architectural style that underpins Web services. Coarse-grained Web services, conversely, are SOA compliant and work at a higher level of abstraction. Each Web service corresponds to a high-level management task (e.g., balancing the load of a virtual resource across multiple physical resources) or a macro-component of a management application (e.g., an event correlator). Coarse-grained Web services are autonomous (each of them fulfills some functionality on its own), loosely coupled with the rest of the management application, and coarse-grained. We distinguished several categories of Web service for integrated management and illustrated them by examples.

In the future, will integrated management adopt the vision of Web services promoted by SOA? Answering this question is difficult because the market is only partially driven by technical considerations. From a business standpoint, adopting Web services makes sense because the software industry is massively adopting this technology, so there are many tools available and many experts in the field. From an engineering standpoint, Web services make it possible to organize management applications differently. This is appealing for architecting

integrated management applications aimed at large organizations. It also opens new horizons regarding the organizational model of standard management architectures: we now have the technology to do without the “manager does everything, agent does nothing” principle. Web services make it possible to break up management platforms into independent macro-components, packaged as services, which can bring more competition among vendors and drive costs down. For instance, one vendor may provide elaborate views with Graphical User Interfaces (GUIs) while another may provide the event correlator. They also facilitate the integration of network, systems and service management tasks. But the performance of coarse-grained Web services for building management solutions is still an open issue that deserves thorough investigations. The same problem was true of CORBA and J2EE a few years ago, and satisfactory solutions appeared a few years later. Can we count on this for Web services too?

ACKNOWLEDGMENTS

The work presented in this paper was supported in part by the EMANICS Network of Excellence (#26854) of the IST Program funded by the European Commission. Early versions of this material were published in [24], [55] and [77].

REFERENCES

- [1] G. Alonso, F. Casati, H. Kuno and V. Machiraju, *Web Services: Concepts, Architectures and Applications*, Springer, 2004.
- [2] A. Alves, A. Arkin, S. Askary *et al.* (Eds.), *Web Services Business Process Execution Language Version 2.0*, Public Review Draft, OASIS, August 2006. Available at: <http://docs.oasis-open.org/wsbpel/2.0/wsbpel-specification-draft.html>
- [3] M. Baldi and G.P. Picco, “Evaluating the Tradeoffs of Mobile Code Design Paradigms in Network Management Applications”, in R. Kemmerer and K. Futatsugi (Eds.), *Proc. 20th International Conference on Software Engineering (ICSE 1998)*, Kyoto, Japan, April 1998, pp. 146–155.
- [4] K. Ballinger, D. Ehnebuske, C. Ferris *et al.* (Eds.), *Basic Profile Version 1.1*, Final Material, Web Services Interoperability Organization, April 2006. Available at <http://www.ws-i.org/Profiles/BasicProfile-1.1-2006-04-10.html>
- [5] T. Banks, *Web Services Resource Framework (WSRF) – Primer v1.2*, Committee Draft 02, May 2006. Available at: <http://docs.oasis-open.org/wsrp/wsrp-primer-1.2-primer-cd-02.pdf>
- [6] T. Bellwood, *UDDI Version 2.04 API Specification*, UDDI Committee Specification, OASIS, July 2002. Available at: <http://uddi.org/pubs/ProgrammersAPI-V2.04-Published-20020719.pdf>
- [7] T. Berners-Lee, J. Hendler and O. Lassila, “The Semantic Web”, *Scientific American*, May 2001.
- [8] K.P. Birman, “Like It or Not, Web Services Are Distributed Objects”, *Communications of the ACM*, Vol. 47, No. 12, pp. 60–62, December 2004.
- [9] D. Booth, H. Haas, F. McCabe *et al.*, (Eds.), *Web Services Architecture*, W3C Working Group Note, World Wide Web Consortium, 2004. Available at: <http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/>

- [10] D. Box, E. Christensen, F. Curbera *et al.*, *Web Services Addressing (WS-Addressing)*, W3C Member Submission, August 2004. Available at: <http://www.w3.org/Submission/2004/SUBM-ws-addressing-20040810/>
- [11] V. Bullard and W. Vambenepe (Eds.), *Web Services Distributed Management: Management Using Web Services (MUWS 1.1) Part 1*, OASIS Standard, August 2006. Available at: <http://docs.oasis-open.org/wsdm/wsdm-muws1-1.1-spec-os-01.htm>
- [12] V. Bullard and W. Vambenepe (Eds.), *Web Services Distributed Management: Management Using Web Services (MUWS 1.1) Part 2*, OASIS Standard, August 2006. Available at: <http://docs.oasis-open.org/wsdm/wsdm-muws2-1.1-spec-os-01.htm>
- [13] B. Burke and R. Monson-Haefel, *Enterprise JavaBeans 3.0*, 5th edition, O'Reilly, 2006.
- [14] N. Catania, P. Kumar, B. Murray *et al.*, *Overview—Web Services Management Framework, Version 2.0*, July 2003. Available at: <http://devresource.hp.com/drc/specifications/wsmf/WSMF-Overview.jsp>
- [15] K. Channabasavaiah, K. Holley and E. Tuggle, “Migrating to a Service-Oriented Architecture, Part 1”, IBM, December 2003. Available at: <http://www-128.ibm.com/developerworks/webservices/library/ws-migratesoa/>
- [16] R. Chinnici, J.J. Moreau, A. Ryman *et al.* (Eds.), *Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language*, W3C Candidate Recommendation, March 2006. Available at: <http://www.w3.org/TR/2006/CR-wsdl20-20060327/>
- [17] R. Chinnici, H. Haas, A.A. Lewis (Eds.), *Web Services Description Language (WSDL) Version 2.0 Part 2: Adjuncts*, W3C Candidate Recommendation, March 2006. Available at: <http://www.w3.org/TR/2006/CR-wsdl20-adjuncts-20060327/>
- [18] M. Choi, J.W. Hong and H. Ju, “XML-Based Network Management for IP Networks”, *ETRI Journal*, Vol. 25, No. 6, pp. 445-463, December 2003.
- [19] M. Choi and J.W. Hong, “Performance Evaluation of XML-based Network Management”, Presentation at the *16th IRTF-NMRG Meeting*, Seoul, Korea, 2004, <http://www.ibr.cs.tu-bs.de/projects/nmrng/meetings/2004/seoul/choi.pdf>
- [20] J. Clark and S. DeRose, *XML Path Language (XPath) Version 1.0*, W3C Recommendation, November 1999. Available at: <http://www.w3.org/TR/1999/REC-xpath-19991116/>
- [21] L. Clement, A. Hatley, C. von Riegen *et al.* (Eds.), *UDDI Version 3.0.2*, UDDI Spec Technical Committee Draft, OASIS, October 2004. Available at: <http://www.oasis-open.org/committees/uddi-spec/doc/spec/v3/uddi-v3.0.2-20041019.htm>
- [22] F. Curbera, Y. Golland, J. Klein *et al.*, “Business Process Execution Language for Web Services (BPEL4WS) Version 1.1”, May 2003. Available at: <http://www-106.ibm.com/developerworks/library/ws-bpel/>
- [23] DMTF, *Web Services for Management (WS-Management)*, Version 1.0.0a, DSP0226, April 2006. Available at: http://www.dmtf.org/standards/published_documents/DSP0226.pdf
- [24] T. Drevers, R. v.d. Meent and A. Pras, “Prototyping Web Services based Network Monitoring”, in J. Harjo, D. Moltchanov and B. Silverajan (Eds.), *Proc. of the 10th Open European Summer School and IFIP WG6.3 Workshop (EUNICE 2004)*, Tampere, Finland, June 2004, pp. 135–142.
- [25] R. v. Engelen, “gSOAP Web services toolkit”, 2003, <http://www.cs.fsu.edu/~engelen/soap.html>
- [26] R. v. Engelen and K.A. Gallivany, “The gSOAP Toolkit for Web Services and Peer-To-Peer Computing Networks”, in *Proc. of the 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid 2002)*, Berlin, Germany, May 2002, pp. 128–135.
- [27] T. Erl, *Service-Oriented Architecture: Concepts and Technology*, Prentice Hall PTR, 2005.

- [28] R.T. Fielding, *Architectural Styles and the Design of Network-Based Software Architectures*, Ph.D. Thesis, University of California, Irvine, CA, USA, 2000. Available at: <http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm>
- [29] R. Fielding, J. Gettys, J. Mogul *et al.* (Eds.), *RFC 2616: Hypertext Transfer Protocol -- HTTP/1.1*, IETF, June 1999.
- [30] S. Graham, A. Karmarkar, J. Mischkinsky *et al.* (Eds.), *Web Services Resource 1.2 (WS-Resource)*, OASIS Standard, 1 April 2006.
- [31] M. Grand, *Patterns in Java, Volume 1: A Catalog of Reusable Design Patterns Illustrated with UML*, Wiley, 1998.
- [32] M. Gudgin, M. Hadley, J.J. Moreau *et al.* (Eds.), *SOAP 1.2 Part 1: Messaging Framework*, W3C Candidate Recommendation, World Wide Web Consortium, 2002. Available at: <http://www.w3.org/TR/soap12-part1/>
- [33] M. Gudgin, M. Hadley, J.J. Moreau *et al.* (Eds.), *SOAP 1.2 Part 2: Adjuncts*, W3C Candidate Recommendation, World Wide Web Consortium, 2002. Available at: <http://www.w3.org/TR/soap12-part2/>
- [34] H. Haas and C. Barreto, "Foundations and Future of Web Services", Tutorial presented at the *3rd IEEE European Conference on Web Services (ECOWS 2005)*, Växjö, Sweden, November 2005.
- [35] H.G. Hegering, S. Abeck and B. Neumair, *Integrated Management of Networked Systems: Concepts, Architectures, and Their Operational Application*, Morgan Kaufmann, 1999.
- [36] M.N. Huhns and M.P. Singh (Eds.), *Readings in Agents*, Morgan Kaufmann, 1998.
- [37] IBM developerWorks, *New to SOA and Web Services*. Available at: <http://www-128.ibm.com/developerworks/webservices/newto/>
- [38] IRTF-NMRG, *Minutes of the 11th IRTF-NMRG Meeting*, Osnabrück, Germany, September 2002. Available at: <http://www.ibr.cs.tu-bs.de/projects/nmr/minutes/minutes-011.txt>
- [39] ISO/IEC 10746-1, *Information Technology—Open Distributed Processing—Reference Model: Overview*, International Standard, December 1998.
- [40] ITU-T, *Recommendation X.753, Information Technology—Open Systems Interconnection—Systems Management: Command Sequencer for Systems Management*, ITU, October 1997.
- [41] ITU-T, *Recommendation M.3400, Maintenance: Telecommunications Management Network—TMN Management Functions*, ITU, February 2000.
- [42] N.R. Jennings and M.J. Wooldridge (Eds.), *Agent Technology: Foundations, Applications, and Markets*, Springer, 1998.
- [43] P. Kalyanasundaram, A.S. Sethi, C.M. Sherwin *et al.*, "A Spreadsheet-Based Scripting Environment for SNMP", in A. Lazar, R. Saracco and R. Stadler (Eds.), *Integrated Network Management V—Proc. of the 5th IFIP/IEEE International Symposium on Integrated Network Management (IM 1997)*, San Diego, CA, USA, May 1997, pp. 752–765.
- [44] J. Kangasharju, S. Tarkoma and K. Raatikainen, "Comparing SOAP Performance for Various Encodings, Protocols, and Connections", in *Proc. of the 8th International Conference on Personal Wireless Communications (PWC 2003)*, Venice, Italy, September 2003, Springer, LNCS, Vol. 2775, pp. 397–406.
- [45] N. Kavantzaz, D. Burdett and G. Ritzinger (Eds.), *Web Services Choreography Description Language Version 1.0*, W3C Candidate Recommendation, World Wide Web Consortium, November 2005. Available at: <http://www.w3.org/TR/2005/CR-ws-cdl-10-20051109/>
- [46] D. Levi and J. Schoenwaelder (Eds.), *RFC 3165: Definitions of Managed Objects for the Delegation of Management Scripts*, IETF, August 2001.
- [47] D. Libes, *Exploring Expect: A Tcl-based Toolkit for Automating Interactive Programs*, O'Reilly, 1994.

- [48] M. Little, "Transactions and Web Services", *Communications of the ACM*, Vol. 46, No. 10, pp. 49–54, October 2003.
- [49] K. Ma, "Web Services: What's Real and What's Not?", *IEEE Professional*, Vol. 7, No. 2, pp. 14–21, March-April 2005.
- [50] C.M. MacKenzie, K. Laskey, F. McCabe *et al.* (Eds.), *Reference Model for Service Oriented Architecture 1.0*, Committee Specification 1, OASIS, August 2006.
- [51] F. Manola and E. Miller, *RDF Primer*, W3C Recommendation, February 2004. Available at: <http://www.w3.org/TR/2004/REC-rdf-primer-20040210/>
- [52] D. Martin, M. Burstein, D. McDermott *et al.*, "Bringing Semantics to Web Services with OWL-S", *World Wide Web: Internet and Web Information Systems*, to appear in 2007.
- [53] J.P. Martin-Flatin, *Web-Based Management of IP Networks and Systems*, Wiley, 2003.
- [54] J.P. Martin-Flatin, "Distributed Event Correlation and Self-Managed Systems", in O. Babaoglu *et al.* (Eds.), *Proc. of the International Workshop on Self-* Properties in Complex Information Systems (Self-Star 2004)*, Bertinoro, Italy, May 2004, pp. 61–64.
- [55] J.P. Martin-Flatin, P.A. Doffoel and M. Jeckle, "Web Services for Integrated Management: a Case Study", in L.J. Zhang and M. Jeckle (Eds.), *Web Services—Proc. of the 2nd European Conference on Web Services (ECOWS 2004)*, Erfurt, Germany, September 2004, Springer, LNCS, Vol. 3250, pp. 239–253.
- [56] K. McCloghrie, D. Perkins, and J. Schoenwaelder (Eds.), *RFC 2578: Structure of Management Information Version 2 (SMIv2)*, IETF, April 1999.
- [57] K. McCloghrie and F. Kastenholtz (Eds.), *RFC 2863: The Interfaces Group MIB*, IETF, June 2000.
- [58] D.L. McGuinness and F. van Harmelen, *OWL Web Ontology Language Overview*, W3C Recommendation, February 2004. Available at: <http://www.w3.org/TR/2004/REC-owl-features-20040210/>
- [59] S. McIlraith, T.C. Son and H. Zeng, "Semantic Web Services", *IEEE Intelligent Systems*, Vol. 16, No. 2, pp. 46–53, March-April 2001.
- [60] A. Nadalin, C. Kaler, R. Monzillo *et al.*, *Web Services Security: SOAP Message Security 1.1 (WS-Security)*, OASIS Standard Specification, February 2006.
- [61] R. Nisse, R. Lemos Vianna, L. Zambenedetti Granville *et al.*, "Implementation and Bandwidth Consumption Evaluation of SNMP to Web Services Gateways", *Proc. of the 9th IEEE/IFIP Network Operations and Management Symposium (NOMS 2004)*, Seoul, Korea, April 2004, pp. 715–728.
- [62] Net-SNMP home page, <http://net-snmp.sourceforge.net/>
- [63] OASIS Web Services Distributed Management (WSDM) Technical Committee, http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsdm
- [64] OASIS Web Services Resource Framework (WSRF) Technical Committee, http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsrf
- [65] OASIS Web Services Transaction (WS-TX) Technical Committee, http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=ws-tx
- [66] L. O'Brien, L. Bass and P. Merson, *Quality Attributes and Service-Oriented Architectures*, Technical Report CMU/SEI-2005-TN-014, Software Engineering Institute, Carnegie Mellon University, PA, USA, September 2005. Available at: <http://www.sei.cmu.edu/pub/documents/05.reports/pdf/05tn014.pdf>
- [67] R. Orfali, D. Harkey and J. Edwards, *Client/Server Survival Guide*, 3rd edition, Wiley, 1999.
- [68] E. O'Tuathail and M. Rose, *RFC 4227: Using the Simple Object Access Protocol (SOAP) in Blocks Extensible Exchange Protocol (BEEP)*, IETF, January 2006.

- [69] C. Pahl, "A Conceptual Framework for Semantic Web Services Development and Deployment", in L.J. Zhang and M. Jeckle (Eds.), *Web Services—Proc. of the 2nd European Conference on Web Services (ECOWS 2004)*, Erfurt, Germany, September 2004, Springer, LNCS, Vol. 3250, pp. 270–284.
- [70] M.P. Papazoglou, "Web Services and Business Transactions", *World Wide Web: Internet and Web Information Systems*, Vol. 6, No. 1, pp. 49–91, March 2003.
- [71] M.P. Papazoglou and D. Georgakopoulos (Eds.), Special issue on "Service-Oriented Computing", *Communications of the ACM*, Vol. 46, No. 10, October 2003.
- [72] M.P. Papazoglou and W.J. van den Heuvel, "Web Services Management: A Survey", *IEEE Internet Computing*, Vol. 9, No. 6, pp. 58–64, November-December 2005.
- [73] S. Parastatidis, S. Woodman, J. Webber *et al.*, "Asynchronous Messaging between Web Services Using SSDL", *IEEE Internet Computing*, Vol. 10, No. 1, pp. 26–39, January-February 2006.
- [74] G. Pavlou, "Chapter 2. OSI Systems Management, Internet SNMP, and ODP/OMG CORBA as Technologies for Telecommunications Network Management", in S. Aidarous and T. Plevyak (Eds.), *Telecommunications Network Management: Technologies and Implementations*, pp. 63–110, IEEE Press, 1998.
- [75] G. Pavlou, P. Flegkas and S. Gouveris, "Performance Evaluation of Web Services as Management Technology", Presentation at the *15th IRTF-NMRG Meeting*, Bremen, Germany, January 2004.
- [76] G. Pavlou, P. Flegkas, S. Gouveris *et al.*, "On Management Technologies and the Potential of Web Services", *IEEE Communications*, Vol. 42, No. 7, July 2004.
- [77] A. Pras, T. Drevers, R. v.d. Meent and D. Quartel, "Comparing the Performance of SNMP and Web Services-Based Management", *IEEE e-Transactions on Network and Service Management*, Vol. 1, No. 2, December 2004.
- [78] J. Rao and X. Su, "A Survey of Automated Web Service Composition Methods", in J. Cardoso and A. Sneth (Eds.), *Proc. of the 1st International Workshop on Semantic Web Services and Web Process Composition (SWSWPC 2004)*, San Diego, CA, USA, July 2004, Springer, Vol. 3387, pp. 43–54.
- [79] J. Schönwälder, A. Pras and J.P. Martin-Flatin, "On the Future of Internet Management Technologies", *IEEE Communications Magazine*, Vol. 41, No. 10, pp. 90–97, October 2003.
- [80] M. Singh and M.N. Huhns, *Service-Oriented Computing: Semantics, Processes, Agents*, Wiley, 2005.
- [81] M. Sloman, "Policy Driven Management for Distributed Systems", *Journal of Network and Systems Management*, Vol. 2, No. 4, pp. 333–360, December 1994.
- [82] M. Sloman and K. Twidle, "Chapter 16. Domains: A Framework for Structuring Management Policy". In M. Sloman (Ed.), *Network and Distributed Systems Management*, Addison-Wesley, 1994, pp. 433–453.
- [83] J. v. Sloten, A. Pras and M.J. v. Sinderen, "On the Standardisation of Web Service Management Operations", in J. Harjo, D. Moltchanov and B. Silverajan (Eds.), *Proc. of the 10th Open European Summer School and IFIP WG6.3 Workshop (EUNICE 2004)*, Tampere, Finland, June 2004, pp. 143–150.
- [84] Soapware.org, <http://www.soapware.org/>
- [85] W. Stallings, *SNMP, SNMPv2, SNMPv3, and RMON 1 and 2*, 3rd Edition, Addison-Wesley, 1999.
- [86] R. Steinmetz and K. Wehrle (Eds.), *Peer-to-Peer Systems and Applications*, Springer, LNCS, Vol. 3485, 2005.
- [87] The Stencil Group, *The Evolution of UDDI—UDDI.org White Paper*, July 2002. Available at: http://www.uddi.org/pubs/the_evolution_of_uddi_20020719.pdf
- [88] UDDI.org, *UDDI Technical White Paper*, September 2000. Available at: http://www.uddi.org/pubs/Iru_UDDI_Technical_White_Paper.pdf

- [89] W. Vogels, “Web Services Are Not Distributed Objects”, *Internet Computing*, Vol. 7, No. 6, pp. 59–66, November-December 2003.
- [90] WASP UDDI, http://www.systinet.com/products/wasp_uddi/overview/
- [91] K. Wilson and I. Sedukhin (Eds.), Web Services Distributed Management: Management of Web Services (WSDM-MOWS) 1.1, OASIS Standard, 01 August 2006. Available at: <http://docs.oasis-open.org/wsdm/wsdm-mows-1.1-spec-os-01.htm>
- [92] Y. Yemini, G. Goldszmidt and S. Yemini, “Network Management by Delegation”, in I. Krishnan and W. Zimmer (Eds.), *Proc. of the 2nd IFIP International Symposium on Integrated Management (ISINM 2001)*, Washington, DC, USA, April 1991, pp. 95–107.
- [93] C. Zhou, L.T. Chia and B.S. Lee, “Semantics in Service Discovery and QoS Measurement”, *IEEE Professional*, Vol. 7, No. 2, pp. 29–34, March-April 2005.
- [94] Zlib home page, <http://www.zlib.org/>

Chapter 1

Internet Management Protocols

Jürgen Schönwälder
Jacobs University Bremen, Germany

Contents

1.1	Introduction	2
1.2	Management Protocol Requirements	2
1.2.1	Configuration Requirements	2
1.2.2	Monitoring Requirements	5
1.2.3	Event Notification Requirements	5
1.2.4	Testing and Troubleshooting Requirements	7
1.2.5	Measurement Data Collection Requirements	7
1.2.6	Security Requirements	8
1.2.7	Non-Functional Requirements	9
1.3	Architectural Concepts	10
1.3.1	Protocol Engines, Functions, and Entities	10
1.3.2	Subsystems and Models	12
1.3.3	Naming and Addressing	12
1.4	Internet Management Protocols	13
1.4.1	Simple Network Management Protocol (SNMP)	13
1.4.2	Network Configuration Protocol (NETCONF)	20
1.4.3	System Logging Protocol (SYSLOG)	24
1.4.4	Flow Information Export Protocol (IPFIX)	29
1.5	Conclusions	34

1.1 Introduction

Operating a large communication network requires tools to assist in the configuration, monitoring, and troubleshooting of network elements such as switches, routers, or firewalls. In addition, it is necessary to collect event reports to identify and track failures or to provide a log of network activities. Finally, it is necessary to collect measurement data for network planning and billing purposes.

With a handful of devices, many of the activities mentioned above can be supported in an ad-hoc approach by using whatever functionality and interface is provided by a given device. However, when the number of network elements and services increases, it becomes essential to implement more advanced management approaches and deploy specific management tools. Since communication networks often consist of a large number of devices from different vendors, the need for standardized management interfaces becomes apparent.

One particularly important management interface for standardization and interoperability is the interface between network elements and their management systems. Network elements are typically closed systems and it is not possible to install management software on the devices them-self. This makes network elements such as routers and switches fundamentally different from general purpose computing systems. Interfaces between management systems have seen less need for standardization since management systems usually run on full featured host systems and it is therefore easier to use general middleware technologies as a means to provide connectivity between management systems.

This chapter surveys Internet management protocols that are mainly used between network elements and their management systems. Since it is impossible to discuss all protocols in detail, the discussion in this chapter will focus on the requirements that have driven the design of the protocols and the architectural aspects through which the protocols address the requirements. By focusing the discussion on requirements and architectural concepts, it becomes possible to document basic principles that are not restricted to the current set of Internet management protocols, but which will likely apply to future management protocols as well.

1.2 Management Protocol Requirements

Before discussing Internet management protocols, it is useful to look at the requirements these protocols should address. The approach taken to identify the requirements is to discuss the main functions management protocols have to support.

1.2.1 Configuration Requirements

Network configuration management has the goal to define and control the behavior of network elements such that a desired network behavior is realized in a

reliable and cost-effective manner. To understand the complexity of this task, it is important to note that the behavior of a network element is not only defined by its configuration, but also by the interaction with other elements in the network. It is therefore necessary to distinguish between *configuration state* and *operational state*. The configuration state is installed by management operations while the operational state is typically installed by control protocols or self-configuration mechanisms.

As an example, consider how the packet forwarding behavior of an IP router is determined. A simple router on a stub network likely has static configuration state in the form of a forwarding table which determines how IP packets are forwarded. This forwarding table is usually established once and subsequently changed manually when the network topology changes. A more complex router on a more dynamic network usually takes part in a routing protocol exchange which is used to compute suitable forwarding tables, taking into account the current state of the network. The forwarding behavior of a router participating in a routing protocol therefore is determined by operational state which was obtained at runtime and influenced by neighboring network elements. Note that the way the router participates in the routing protocol exchange is likely determined itself by configuration state, which sets the parameters and constraints under which shortest paths are computed.

Requirement 1.1 *A configuration management protocol must be able to distinguish between configuration state and operational state.*

The example above introduces another important aspect in network configuration management: since devices typically exchange control information to establish operational state, configuration state changes on a single device can affect the operational state in many other devices. As a consequence, configuration state changes can have side effects that may affect a whole network.

With this in mind, it becomes clear that it is not sufficient to look solely at the configuration state of a device in order to identify why a certain network element does not behave as desired. To understand why an element misbehaves, it is necessary to look at the combined configuration state and operational state and it is desirable that devices report configuration information in a way which allows to distinguish between configuration state and operational state. In addition, it is desirable that the device also records sufficient information which allows an operator to figure out why and when operational state was established. For completeness, it should be noted that network elements maintain statistics in addition to configuration and operations state.

Configuration changes in a network may originate from different systems. In addition, it might be necessary to change several systems for a logical change to be complete. This leads to the following two requirements:

Requirement 1.2 *A configuration management protocol must provide primitives to prevent errors due to concurrent configuration changes.*

Requirement 1.3 *A configuration management protocol must provide primitives to apply configuration changes to a set of network elements in a robust and transaction-oriented way.*

Requirement 1.4 *It is important to distinguish between the distribution of configurations and the activation of a certain configuration. Devices should be able to hold multiple configurations.*

Devices usually maintain different configurations. It is common to distinguish between the currently running configuration and the configuration that should be loaded at startup time. Some devices also support an open ended collection of different configurations. This leads to the following two requirements:

Requirement 1.5 *A configuration management protocol must be able to distinguish between several configurations.*

Requirement 1.6 *A configurations management protocol must be clear about the persistence of configuration changes.*

Since different management systems and operators may have access to the configuration of a device, it is important to log information that can be used to trace when and why a certain configuration change has been made.

Requirement 1.7 *A configuration management protocol must be able to report configuration change events to help tracing back configuration changes.*

Not all devices can be expected to have the resources necessary to track all configuration changes throughout their lifetime. It is therefore common practice to keep copies of device configurations on host computers and it is thus desirable to be able to use standard version control systems to track changes.

Requirement 1.8 *A full configuration dump and a full configuration restore are primitive operations frequently used by operators and must be supported appropriately.*

Requirement 1.9 *A configuration management protocol must represent configuration state and operational state in a form which allows to use existing standard comparison and versioning tools.*

Network operators are generally interested in providing smooth services. The number of service interruptions must be minimized and stability should be maintained as much as possible. A smart device should therefore be able to load a new configuration without requiring a full device reset or the loss of all operational state.

Requirement 1.10 *Configurations must be described such that devices can determine a set of operations to bring the devices from a given configuration state to the desired configuration state, minimizing the impact caused by the configuration change itself on networks and systems.*

1.2.2 Monitoring Requirements

After configuring the devices making up a network, it is essential to monitor that they are functioning correctly and to collect some basic usage statistics such as link utilizations. The setup of monitoring tools can be greatly simplified if devices reveal their capabilities.

Requirement 1.11 *Monitoring protocols should support the discovery of the capabilities of a device.*

Networks may contain a large number of manageable devices. Over time, monitoring requirements also tend to increase in terms of the number of data items to be monitored per device. As a consequence, monitoring protocols should be highly scalable.

Requirement 1.12 *Monitoring protocols must scale to a large number of devices as well as a large number of data items to be monitored.*

Devices often contain large numbers of similar counters and in many cases not all of them are relevant for monitoring purposes. It is therefore crucial to be able to select which subsets of management data need to be retrieved for monitoring purposes, sometimes up to the level of an individual counter.

Requirement 1.13 *It must be possible to perform monitoring operations on selected subsets of management data.*

Requirement 1.14 *Monitoring protocols must support a naming scheme which is capable to identify instances up to the granularity of a specific counter.*

Finally, it should be noted that devices are built for some primary function (e.g., a router is built primarily to forward IP packets) and monitoring functions should thus have little negative impact on the primary functions of a device.

Requirement 1.15 *Monitoring protocols should have low impact on the primary functions of a device. In particular, it must be possible that multiple applications monitor different aspects of a single device without undue performance penalties for the device.*

1.2.3 Event Notification Requirements

Operationally relevant status changes and in particular failures typically happen asynchronously. Devices therefore need a mechanism to report such events in a timely manner. As will be seen later, there are multiple protocols to transport event notifications. Regardless of the protocol, it is important to be able to identify some key parameters about an event

Requirement 1.16 *Event notifications must have sufficient information to identify the event source, the time the event occurred, the part of the system that originated the event notification, and a broad severity classification.*

Since event notifications may trigger complex procedures or work-flows, it is necessary to ensure the integrity and authenticity of an event notification. In some cases, it is even required to prove the event integrity of events stored in a permanent event log.

Requirement 1.17 *It is desirable to be able to verify the integrity of event notifications and the authenticity of the event source.*

Events may originate from many different devices and processes in a network and it is likely that formats and protocols are not always consistent. Furthermore, event notifications may have to pass middleboxes such as firewalls and network address translators and there might be multiple levels of filtering and different data collectors for different purposes.

Requirement 1.18 *Event notifications may pass through a chain of relays and collectors which should be (a) transparent for the event notification originator and (b) not change the content of event notifications in any way.*

One particular design issue that needs to be addressed when designing an event notification system are effective mechanisms to deal with so called event storms. Event storms happen for example after a power outage or a cut of a fiber when multiple devices and processes detect an error and issue event notifications. Event storms look very similar to denial of service attacks, except that the event messages actually originate from a valid source.

Requirement 1.19 *Notification senders should provide effective throttling mechanisms in order to deal with notification storms.*

A reliable event notification transport is often desirable. However, it should be noted that choosing a reliable transport layer protocol does not by itself provide a reliable notification delivery system since the transport might simply fail while the network is under stress or attack or the notification receiver might not understand the notification. For a highly reliable event notification system, it is therefore essential to provide a local notification logging facility and a communication protocol providing confirmed delivery of event notifications.

Requirement 1.20 *Highly reliable event notification systems must provide confirmed event notification protocols and logging facilities to store event notifications at the event source.*

Finally, it has proven to be valuable that event notifications can be read by both machines and humans.

Requirement 1.21 *Event notifications should include machine readable structured data as well as human readable event descriptions.*

1.2.4 Testing and Troubleshooting Requirements

Troubleshooting and testing is a common activity, especially when new services and networks are deployed. Management protocols should support these activities.

Requirement 1.22 *A management protocol must allow the invocation of test functions for trouble shooting purposes.*

The ability to invoke test functions from different locations in a given network topology can lead to specific insights about the root cause of observed problems very quickly. In many situations, it is desirable to execute test functions periodically so that alarms can be triggered if tests fail or an expected heartbeat message is not received.

Requirement 1.23 *Scheduling functions should be provided to execute testing functions periodically or following a calendar schedule.*

1.2.5 Measurement Data Collection Requirements

It is important to collect accurate measurement data sets describing how networks are being used. Such information is, among other things, useful for usage-based accounting, network planning and traffic engineering, quality of service monitoring, trouble shooting, or attack and intrusion detection .

To collect measurement data, it is necessary to define observations points where meters are attached to a network in order to collect measurement data sets. These data sets are then exported by an exporting process, which transfers the data sets to a collecting process.

There are several specific measurement data collection requirements. The first one deals with scalability:

Requirement 1.24 *Measurement data collection protocols must scale to hundreds of exporting processes. Furthermore, data exporting processes must be able to export to multiple data collecting processes.*

Scalability and efficiency is a key requirement. Since network bandwidth tends to grow faster than processor speeds, it can be expected that only statistical sampling techniques can be deployed on very high-speed optical backbone networks in the future. Related to scalability is the requirement for measurement data transfer protocol to be congestion aware.

Requirement 1.25 *Measurement data transfer protocols must be reliable and congestion aware.*

Congestion during the data transfer may lead to significant data loss and may affect other network services. Data loss can, however, occur at many places, namely at the metering process, the export process, during the data transfer, or at the collecting process. To provide a complete picture, it is therefore necessary properly track all data losses.

Requirement 1.26 *It must be possible to obtain an accurate indication about lost measurement data records.*

Since metering processes often attach timestamps to data sets, it is necessary to select timestamps with a sufficiently high granularity and to synchronize the clocks on the meters.

Requirement 1.27 *Metering processes must be able to assign timestamps with a suitable resolution. Furthermore, metering processes must be able to synchronize their clocks with the necessary precision.*

Early protocols for exchanging measurement data sets assumed a specific hardwired data model. This approach has proven problematic several times, leading to the following requirement:

Requirement 1.28 *The data format used by the data transfer model must be extensible.*

Finally, it should be noted that measurement data sets and protocols that transfer them need proper protection.

Requirement 1.29 *Measurement data transfer protocols must provide data integrity, authenticity of the data transferred from an exporting process to a collecting process, and confidentiality of the measurement data.*

For additional details on measurement data collection requirements, see RFC 3917 [43].

1.2.6 Security Requirements

Network management information in general is sensitive. This seems obvious in the context of configuration, but may be less obvious in the context of monitoring. But as security attacks are getting more sophisticated, it is important to increase the protection of data that can identify ongoing attacks since attackers have a natural desire to hide their activities.

Requirement 1.30 *There is a need for secure data transport, authentication, and access control.*

Operationally, it seems that the usage of cryptographically strong security mechanisms do not cause too many problems if the required processing power is available. However, the required key management can be time consuming. Hence, it is important that key and credential management functions integrate well.

Requirement 1.31 *Any required key and credential management functions should integrate well with existing key and credential management infrastructures.*

Once you have authenticated principals, it is natural to apply access control to limit the accessible information to the subset needed to perform a specific function.

Requirement 1.32 *The granularity of access control must match operational needs. Typical requirements are a role-based access control model and the principle of least privilege, where a user can be given only the minimum access necessary to perform a required task.*

Unfortunately, it is often difficult to define precisely which information needs to be accessible for a given management application to function correctly. Many applications resort to some probing and discovery algorithms and they use best guess heuristics in cases where some information is not directly accessible.

1.2.7 Non-Functional Requirements

Several non-functional requirements should be taken into account as well. The first one concerns human readability of protocol messages. The experience with SNMP (which uses binary encoded messages) tells us that the usage of specially encoded messages significantly increases the time programmers spend in writing new applications.

Requirement 1.33 *Human readable protocol messages significantly reduce integration efforts.*

The availability of C/C++/Java libraries with well defined stable APIs does not seem to make it easy enough for developers and operators to create new applications quickly. For network management purposes, good integration into high-level scripting languages is therefore an essential requirement.

Requirement 1.34 *Management protocols should be easy to access from high-level scripting languages.*

On managed devices, it has been observed that the development of management interfaces sometimes happens in a relatively uncoordinated way. This leads often to duplicate implementation and maintenance efforts which increase costs and and inconsistencies across management interfaces which decrease the operator satisfaction.

Requirement 1.35 *Implementation costs on network devices must be minimized and duplicated implementation efforts for different management protocols should be avoided.*

Management systems usually contain a significant amount of code that is specific to a certain operator or network type. Furthermore, many network management tasks involve work-flows (e.g., the ordering of repair parts or a temporary allocation of bandwidth from other providers) that need some level of information technology support. Management applications and protocol interfaces should therefore be easy to integrate with standard software tools.

Requirement 1.36 *Management protocol interfaces should integrate well with standard tools such as revision management tools, text analysis tools, report generation tools, trouble ticketing tools, databases, or in general work-flow support tools.*

1.3 Architectural Concepts

Before discussing concrete protocols, it is useful to introduce some architectural concepts that are being used later in this chapter to describe various Internet management protocol frameworks.

1.3.1 Protocol Engines, Functions, and Entities

The architecture of a management protocol can usually be divided in a *protocol engine* handling protocol messages and *protocol functions* that realize the protocol's functionality. The protocol engine and the protocol functions together form a *protocol entity*, as shown in Figure 1.1.

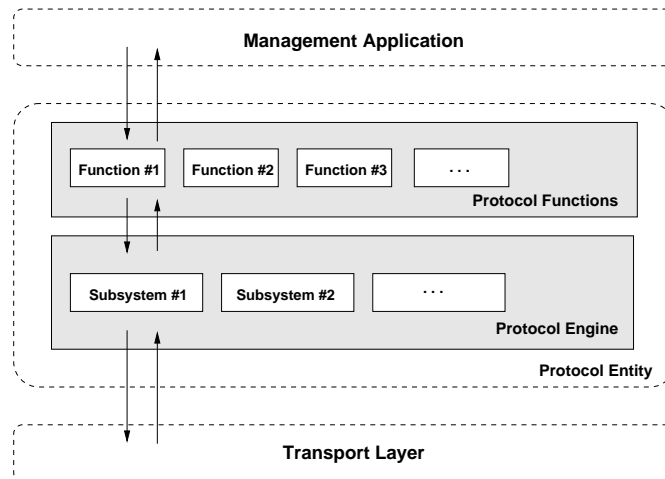


Figure 1.1: Protocol Engines, Functions, and Entities

Protocol Engine

The protocol engine is primarily concerned with the handling of the message formats used by a protocol. New protocols usually have a single message format. But during the design of a new protocol, it must already be anticipated that message formats need changes over time if a protocol is successfully deployed and new requirements must be addressed. Some of the older Internet management protocols have already been gone through this evolution while some of the newer protocols are too young to have experienced a need for different message formats.

From an architectural point of view, however, it is important to plan for message format changes that should be handled by the protocol engine.

An important service provided by the protocol engine is the identification of the principal invoking a protocol function. This usually requires cryptographic mechanisms and hence the protocol engine must deal with message security processing and provide mechanisms to ensure message integrity, data origin authentication, replay protection and privacy. There are two commonly used approaches to achieve this:

1. The first approach is to design specific security processing features into the protocol engine. This has the advantage that the protocol is self-contained and reduces its dependency on external specifications and infrastructure. The downside is that the specification becomes more complex and more difficult to review and that implementation and testing costs increase.
2. The second approach, which is becoming increasingly popular, is the usage of secure transport mechanism. The protocol engine then simply passes authenticated identities through the protocol engine. This has the advantage that prior work on the specification, testing and implementation of security protocols can be leveraged easily.

A protocol engine can also provide access control services to the protocol functions. While the actual call to check access control rules might originate from protocol functions realized outside of the engine, it is usually desirable to maintain the access control decision logic within the protocol engine to ensure that different protocol functions realize a consistent access control policy.

Finally, a protocol engine has to deal with the mapping of the protocol messages to underlying message transports. It seems that in the network management space, it has become increasingly popular to define different transports mappings for a given management protocol and to mark one of them as mandatory to implement to achieve a baseline of interoperability.

Protocol Functions

The protocol functions realize the functionality of a management protocol. A protocol engine usually determines the protocol operation that should be invoked after parsing an incoming message and then searches for a protocol function that has registered to handle the protocol operation. Such a registration can be conceptual and rather static in implementations or it can be something which is dynamic and runtime extensible.

Management protocols differ quite a bit in the number of protocol functions they support. In some management protocols, the number of protocol functions is rather fixed and there is strong consensus that adding new protocol functions should be avoided while other protocols are designed to be extensible at the protocol functions layer.

1.3.2 Subsystems and Models

The internals of a protocol engine can be decomposed into a set of *subsystems*. A subsystem provides a well defined functionality which is accessible through a defined subsystem interface. By documenting the subsystems and their interfaces, the data flow between the components of a protocol engine can be understood. Furthermore, defined subsystem interfaces reduce the number of surprises when a protocol design gets extended over time.

Within a subsystem, there might be one or multiple *models* that implement the subsystem's interface. Subsystems designed to support multiple models make an architecture extensible. However, there are also subsystems that by design do not support multiple models since they for example organize the data flow between subsystems and models.

1.3.3 Naming and Addressing

Management protocols manipulate managed objects, which are abstractions of real resources. The term “managed object” has to be understood in a broad sense; it does not imply that a managed object is an object-oriented data structure. A managed object can be as simple as a simple typed variable or a part in a complex hierarchically structured document.

A particularly important architectural design decision is the *naming*, that is the identification of managed objects. There are several different approaches to name managed objects and the power, flexibility, or simplicity of the naming system has direct impact on the implementation costs and the runtime costs of a management protocol. As a consequence, the selection of a naming mechanism should be treated as an important architectural design decision.

The names of managed objects are usually implicitly scoped. Within the scope, a name provides a unique identifier for a managed object. One commonly used approach is to scope the names relative to the protocol engine that provides access to the managed object. This means that a globally scoped name within a management domain can be constructed out of the identity of the protocol engine and the name of the managed object. Hence, it is important to define how protocol engines are addressed. Some management protocols introduce their own protocol engine addressing scheme while other management protocols use transport endpoint addresses as a shortcut to identify the protocol engine. Architecturally, the first approach is clearly preferable since it decouples the protocol engine from the transport endpoints. Practically, it turns out that such a decoupling has its price since an additional mapping is needed. The selection of a suitable protocol engine addressing scheme therefore becomes an engineering trade-off between architectural purity and implementation and deployment costs.

1.4 Internet Management Protocols

This section describes several management protocols that were specifically designed for managing the Internet. The protocols discussed below are all specified and standardized by the Internet Engineering Task Force (IETF).

1.4.1 Simple Network Management Protocol (SNMP)

The core Internet architecture was developed and tested in the late 1970s by a small group of researchers [11]. During the early 1980s, several important features such as the domain name system were added to the architecture [12]. At the end of the 1980s, the Internet had grown to a world-wide network connecting all major universities and research institutions. Network management functions were at this time realized on an ad-hoc basis through explicit management.

In 1988, it became apparent that additional technology is needed in order to provide better tools to operate the growing Internet infrastructure [8]. The debate held in 1988 shows a fundamental principle in the IETF approach: The IETF seeks to make pragmatic decisions in order to address short-term technical problems. Rather than spending much time on architectural discussions, concrete proposals which are backed up by interoperable implementations have a good chance to be adopted by the IETF and finally in the market place.

The Simple Network Management Protocol (SNMP) [5], the Structure of Management Information (SMI) [47, 48, 46] and an initial Management Information Base (MIB-II) [37] were standardized in 1990 and 1991. One of the fundamental goals of the original SNMP framework was to minimize the number and complexity of management functions realized by the management agent itself [5]. The desire to minimize the complexity of management functions in the network elements reflects the constraints on processing power available in the early 1990s and the hardware/software architecture used in most networking devices at that time. Much has changed in the last 10 years in this aspect. Current network elements, such as routers and switches, implement all the primary and time critical network functions in hardware. Furthermore, the processing speed of the microprocessors embedded in networking devices has increased in magnitudes during the last ten years. But it is also important to understand that the network traffic handled by these devices did evolve exponentially. The question whether today's and tomorrow's network elements have the computational power to realize much more complex management functions is not easy to answer.

The evolution of the original SNMP framework and the MIB modules during the last two decades did provide security, some efficiency improvements, and enhancements of the data definition language. Today, there are more than 100 MIB modules on the standardization track within the IETF and there is an even larger and growing number of enterprise-specific MIB modules defined unilaterally by various vendors, research groups, consortia, and the like resulting in an unknown and virtually uncountable number of defined objects [7, 52].

In December 2002, the specification of SNMP version 3 (SNMPv3) was published as an Internet Standard [25, 6, 30, 1, 58, 41, 40], retiring all earlier versions of the SNMP protocol. The second and current version of the SMI (SMIv2) was already published in April 1999 as an Internet Standard [35, 36, 34].

Deployment Scenarios

A typical SNMP deployment scenario is shown in Figure 1.2. It shows a single manager which is talking SNMP to three agents running on different SNMP-enabled devices.

For one device, a so called SNMP proxy is involved. There can be different reasons for deploying proxies. One reason can be firewalls and in particular network address translators [44]. Another reason can be version mismatches between management stations and devices¹.

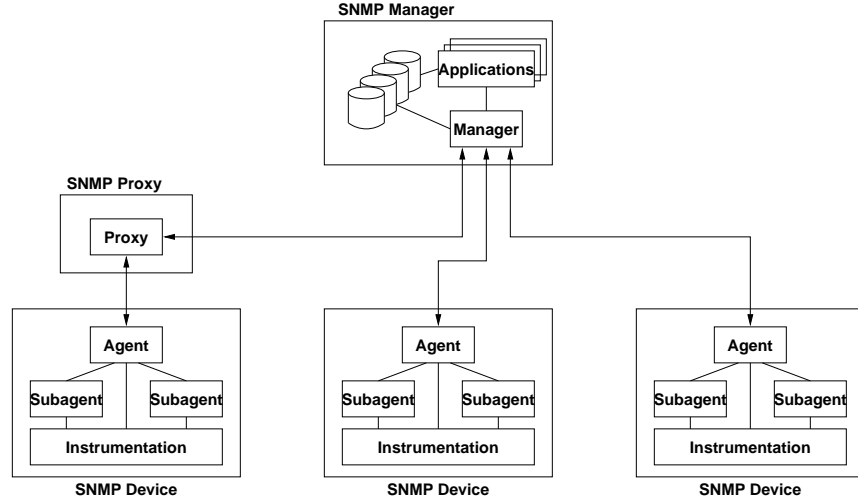


Figure 1.2: SNMP Deployment Scenario

Agents internally often have a master/subagent structure [17]. Subagents can be embedded into line-cards on routers or application processes to provide access to specific management information. The access to management is usually implemented by so called method functions in the instrumentation.

Many management systems use full fledged database management systems to store management information retrieved from network elements. Applications access the data stored in the database to realize management functions. In addition, they may interact directly with devices by using services provided by the SNMP manager interface.

¹Devices have picked up support for SNMPv3 much faster than some widely used management applications.

It should be noted that Figure 1.2 provides a rather traditional view. There are many situations where the notion of an agent and a manager is broken apart. In particular, it is not unusual to logically separate the forwarding and processing of notifications from the initiation and processing of commands to retrieve or modify data stored on SNMP-enabled devices.

Architecture

One of the biggest achievements of the SNMPv3 framework next to the technical improvements of the protocol itself is the architectural model defined in RFC 3411 [25]. It decomposes an SNMP protocol engine into several subsystems and it defines abstract service interfaces between these subsystems. A subsystem can have multiple concrete models where each model implements the subsystem interface defined by the abstract service interfaces. The architectural model thus provides a framework for future SNMP enhancements and extensions.

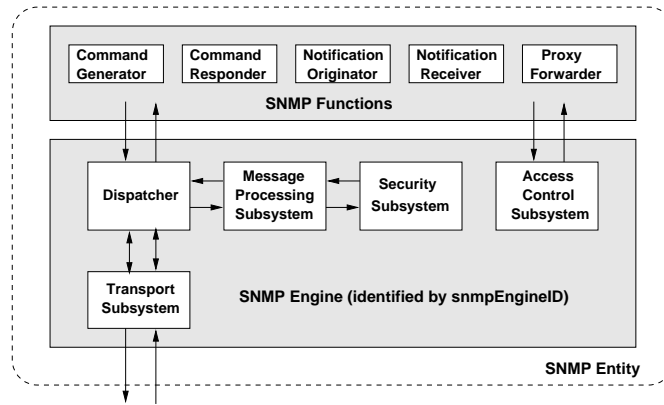


Figure 1.3: Subsystem Architecture of SNMP

Figure 1.3 shows the decomposition of an SNMP protocol engine into subsystems.² SNMP protocol messages enter an SNMP engine via the transport subsystem. Several different transport models can co-exist in the transport subsystem. The most commonly used transport model is the SNMP over UDP transport model. The transport subsystem passes messages to the dispatcher. The dispatcher, which is a singleton, coordinates the processing in an SNMP engine. It passes incoming messages to the appropriate message processing model in the message processing subsystem. The message processing models take care of the different headers used by the different versions of the SNMP protocol. The message processing model passes messages to a security model in the security subsystem which provides security services such as authentication and privacy of messages. After returning from the security and message processing

²Figure 1.3 does not precisely follow the standard architecture since it includes a transport subsystem which was just recently added to the architectural model [26].

subsystems, incoming messages are passed by the dispatcher to SNMP functions which implement the requested protocol operations. SNMP functions call the access control subsystem which can provide several different access control models for access control services.

The SNMP framework traditionally follows a variable-oriented approach. Management information is represented by collections of variables. Every variable has a simple primitive type. Variables can appear as simple scalars or they can be organized into so-called conceptual tables. Due to this very granular approach, every variable needs a unique name. The current SNMP architecture uses a four level naming system. The name of a variable consists of the following components:

1. The identification of the engine of the SNMP entity which hosts the variable. This is called the *context engine identifier*.
2. The identification of the context within an engine which hosts the variable. This is called the *context name*.
3. The identification of the variable type, also commonly called an *object type*. An object type is identified by means of a registered object identifier.
4. The identification of the variable instance, also called an *instance identifier*. An instance identifier is an object identifier suffix which gets appended to the object identifier of the object type.

Like for every naming system, there needs to be a mechanism to lookup names and to explore the name space. The SNMP framework addresses this requirement by restricting protocol operations to a single context. A context is identified by a context engine identifier and a context name. Within the selected context, multiple variables can be read, written, or submitted as part of an event notification.

To explore the name space, SNMP uses a two-level approach:

- Contexts are typically discovered through discovery procedures. SNMP engines can be discovered by probing their transport endpoints. Once a transport endpoint has been found, SNMP messages can be sent to the SNMP engine listening on that transport endpoint to discover the engine identifier and existing contexts.
- The variables within a context are discovered by using an iterator which traverses the name space while retrieving variables.

Using an iterator, which retrieves data while traversing the name space, has significant advantages in highly dynamic situations. Management Protocols which separate name space traversal operations from data retrieval operations usually are less scalable due to the imposed overhead of managing the name space and synchronization issues in highly dynamic situations.

Data Modeling

Data modeling in the SNMP framework is centered around the notion of conceptual tables [35, 36, 34]. All columns of a conceptual table have a primitive data type. So called textual conventions are used to associate special semantics with a primitive type; textual conventions are roughly comparable to typedefs in languages such as C.

Some columns of a conceptual table are marked as index columns. The values of the index columns uniquely identify a row and the index columns, or short the index, thus serve as the primary key for a conceptual table.

The basic concepts resemble relational databases. However, the SNMP framework puts several constraints on the indexing scheme. For example, conceptual tables can only have one index (key) that can be used for fast lookups and the index is subject to several additional constraints imposed by the protocol itself (index size constraints) or by access control mechanisms.

The data modeling framework supports scalars in addition to conceptual tables. For the discussion in this document, it is however sufficient to consider scalars just a degenerate form of a table which has only one row and a fixed index.

In order to assign tables a unique name, an object identifier naming system is used. Object identifier, a primitive ASN.1 data type, essentially denote a paths in the global object identifier registration tree. The tree has a virtual root and can be used to uniquely assign names to arbitrary objects [54, 19]. In the SNMP framework, tables as well as the table's columns are registered in the tree. Note that the definition of index columns and the registration of tables happen at design time and cannot be changed once a specification has been published. As a consequence, the design of suitable indexing schemes is an important design step in any SNMP data modeling effort.

To fully understand the SNMP framework, it is crucial to properly distinguish between conceptual tables used at the data modeling layer and the registration tree used purely for naming purposes. As the next section explains, the actual protocol operations do neither understand tables nor registration trees and simply operates on an ordered list of variables.

Protocol Operations

The SNMP protocol provides a very small number of protocol operations, which is essentially unchanged since 1993 [41]. The protocol operations can be classified into write operations (**set**), read operations (**get**, **getnext**, **getbulk**), and notify operations (**trap**, **inform**). Figure 1.4 gives an overview which SNMP functions invoke the various operations and which ones are confirmed via a response.

The protocol operations do not operate on conceptual tables nor do they operate on the registration tree. Instead, they solely operate on an ordered collection of variables. The **get**, **set**, **trap**, and **inform** operations all identify variables by their full name (object type identifier plus instance identifier). The

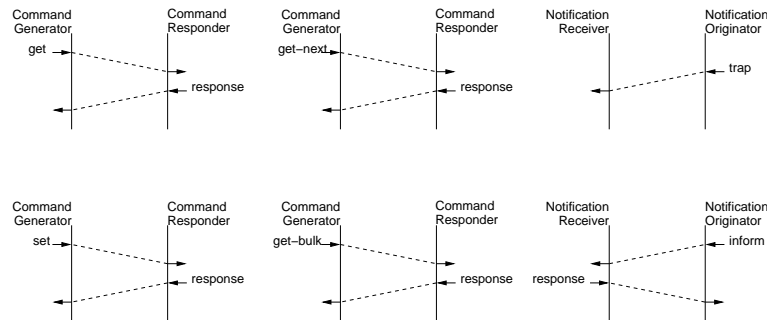


Figure 1.4: Protocol operations of SNMP

getnext and **getbulk** operations obtain the next variable (**getnext**) or variables (**getbulk**) according to lexicographic ordering, which essentially maps the hierarchical name space into a flat ordered collection of variables.

The **getnext** and **getbulk** operations combine the traversal of the variable name space with actual data retrieval. This implies that variables can dynamically come and go without any need to register them in a name service.

Something that is often overlooked is the handling of errors and exceptions. SNMP protocol operations can fail with an error code which indicates the reason of the failure and which variable caused the failure. In the second version of the SNMP protocol operations, exceptions were introduced to report, on a per variable basis, exceptional conditions without failing the whole protocol operation. Not surprising, most error codes deal with the execution of write operations. However, exceptions are also important for read operations to indicate that specific instances do not exist or that an iterator such as **getnext** or **getbulk** has reached the end of the ordered collection of variables.

Security

A great deal of attention has been given to the security of SNMP as the standardization of a secure version of SNMP proved to be very difficult and time consuming. The SNMPv3 architecture has been designed with a message-based security model in mind where every individual message can be authenticated and also encrypted if so desired. The User-based Security Model (USM) [1] protects against the following threats:

- Modification of Information
- Masquerade
- Disclosure
- Message Stream Modification

It does not defend against denial of service attacks. The USM authenticates a user, identified by the user name, who is invoking a read or write operation or

who is receiving a notification. The user name is mapped into a model independent security name which is passed along inside the SNMP engine together with a security level and the identification of the security model.

The triple of security name, security level, and security model is used by the access control subsystem to decide whether an access to a variable is allowed or denied. The View-based Access Control Model (VACM) [58], the only access control model defined so far in the access control subsystem, provides means to define read/write/notify views on the collection of variables and to restrict access to these views.

One of the original design principles for SNMP was to keep SNMP as independent as possible of other network services so that SNMP can continue to function if the network is not running well. The original security mechanism follow this principle and thus come with an SNMP specific user, key, and access control rule management interface. Operators have reported that adding a new user and key management infrastructure has significant costs and that they prefer an approach where security better integrates with existing user, key, and access control management infrastructures. The IETF is at the time of this writing working on an additional security model, which utilizes secure session-based transports such as Secure Shell (SSH) [59] or Transport Layer Security (TLS) [18] protocol [24].

Discussion

The SNMP protocol has been very successful. It is widely deployed in network elements and management systems. However, the main usage of SNMP seems to be monitoring, discovery, and event notification.

SNMP has seen less usage for controlling and configuring devices. By looking at the configuration requirements detailed in Section 1.2.1, it becomes obvious why. SNMP fails to address most of the configuration related requirements and this is perhaps part of the reason why configuration requirements are so well documented in the RFC series [50, 51].

SNMP works well for polling large numbers of devices for a relatively small set of variables. Once the set of variables to be collected increase, SNMP becomes less efficient in terms of required bandwidth but also in terms of overall latency and processing overhead compared to other approaches. The reason is that protocol operations “single step” through the data retrieval, which makes efficient data retrieval and caching on a managed device complicated.

SNMP enjoys significant usage for event notification. However, the binary format of SNMP messages combined with the object identifier naming scheme makes the processing of such event records often difficult in practice and other event notification and logging protocols such as SYSLOG are therefore popular as well.

There were attempts to realize more generic and powerful management services on top of SNMP to delegate management functions [31, 32] or to invoke the remote execution of troubleshooting procedures [32]. These approaches have not seen much uptake since the invocation of remote management procedures

with read/write/notify primitives is rather unnatural and has relatively high development costs.

1.4.2 Network Configuration Protocol (NETCONF)

In 2002, the Internet Architecture Board (IAB) organized a workshop in order to guide the focus of the network management work done in the IETF. The workshop was attended by network operators and protocol developers and resulted into some concrete recommendations [51]. One of the recommendations was to focus resources on the standardization of configuration management mechanisms. Another recommendation was to use the Extensible Markup Language (XML) [2] for data encoding purposes. Some of the discussions related to the future directions are also summarized in [53].

In 2003, the a working group was formed in the IETF to produce a protocol suitable for network configuration. The working group charter mandated that the Extensible Markup Language (XML) [2] be used for data encoding purposes. The protocol produced by this working group is called NETCONF [20] and the subject of this section.

Deployment Scenarios

The driving force behind NETCONF is the need for a programmatic interface to manipulate configuration state. The automation of command line interfaces (CLIs) using programs and scripts has proven to be problematic, especially when it comes to maintenance and versioning issues. Operators have reported that it is time consuming and error prone to maintain programs or scripts that interface with different versions of a command line interface.

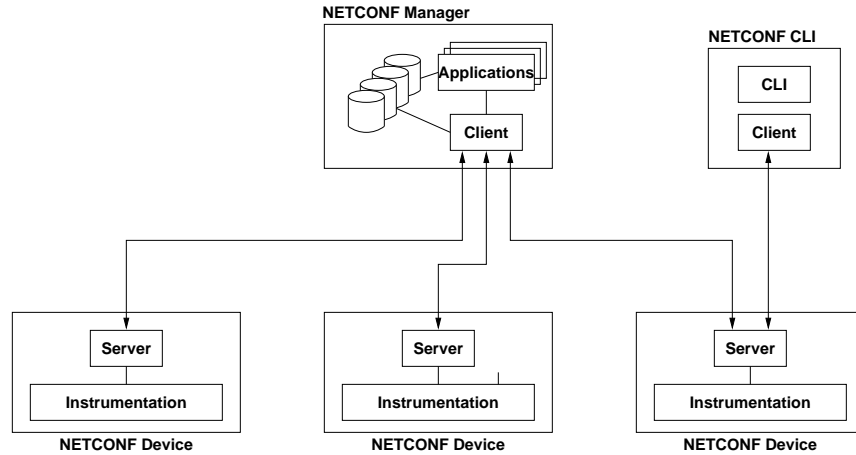


Figure 1.5: NETCONF Deployment Scenario

Figure 1.5 shows a NETCONF deployment scenario. It is expected that

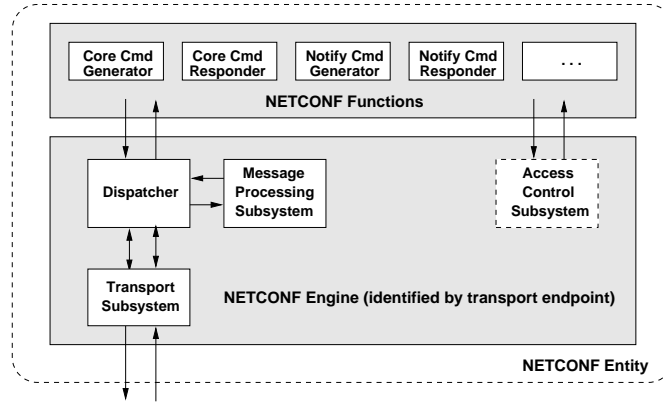


Figure 1.6: Subsystem Architecture of NETCONF

network-wide configuration or policy systems will use the NETCONF protocol to enforce configuration changes. In a policy framework, the manager involving a NETCONF client might act as a policy decision point while a device involving a NETCONF server might act as a policy enforcement point. Of course, such a setup requires that a policy manager can translate higher-level policies into device configurations; NETCONF only provides the protocol to communicate configuration data.

The right part of Figure 1.5 shows a CLI which talks NETCONF to a server in order to implement its functionality. It is important to realize that NETCONF must be powerful enough to drive CLIs. Real cost savings can only be achieved if there is a single method to effect configuration changes in a device which can be shared across programmatic and human operator interfaces. This implies that the scope of the NETCONF protocol is actually much broader than just device configuration.

Architecture

The NETCONF protocol does not have a very detailed architectural model. The protocol specification [20] describes a simple layered architecture which consists of a transport protocol layer, a remote procedure call (RPC) layer, an operation layer providing well defined RPC calls, and a content layer.

A more structured view of NETCONF, which resembles the SNMP architectural model, is shown in Figure 1.6. The transport subsystem provides several transport models. The working group so far has defined an SSH transport model [57], a BEEP transport model [28], and a SOAP transport model [23]. The SSH transport is mandatory to implement.

The dispatcher is the central component which organizes the data flow within the NETCONF engine. It hands messages received from the transport subsystem to the message processing subsystem which currently has only one message processing model. This message processing model handles a capabilities ex-

change, which happens after a transport has been established, and the framing or RPC messages. The operations being invoked are provided by NETCONF functions. The base NETCONF specification deals mainly with generic operations to retrieve and modify configuration state. An additional document defines operations to subscribe to notification channels and to receive notifications. It is expected that additional operations will be introduced in the future for more specific management purposes.

The base NETCONF specification follows a document-oriented approach. The configuration state of a device is considered to be a structured document that can be retrieved and manipulated. A filtering mechanism has been defined which allows to retrieve a subset of the document representing a configuration.

NETCONF supports multiple configuration datastores. A configuration datastore contains all information needed to get a device from its initial default state into the desired configuration state. The **running** datastore is always present and describes the currently active configuration. In addition, NETCONF supports the notion of a **startup** configuration datastore which is loaded at next re-initialization time and a **candidate** datastore which is a scratch buffer which can be manipulated and later committed to the **running** datastore.

Data Modeling

Since NETCONF uses XML to encode network management data and in particular configuration state, it seems obvious to use some XML schema notation to formally specify the format of these XML documents. However, there is no agreement yet which schema language to use. Furthermore, it seems necessary to incorporate additional information in data models that go beyond the capabilities of schema languages or requires to resort to hooks in schema languages where additional information can be specified which is ignored by generic tools.

During the development of the NETCONF protocol specifications, which are formally defined using XML schema [21], it was observed that the XML schema notation is difficult to read and verify for humans. Other schema notations such as Relax NG [15] and especially its compact notation [13] seem to be easier to read and write. Still, these schema languages tend to be relatively far away from an implementors view and in the light of integration with SNMP, it might also be possible to adapt a language such as SMIng [55] for NETCONF data modeling purposes.

The NETCONF working group was not chartered to address the data modeling aspects of NETCONF and as such there is no agreement yet how to approach this problem. It is, however, clear that agreement must be reached on how to write data models in order to achieve standardized configuration in a heterogeneous environment.

Protocol Operations

The NETCONF protocol has a richer set of protocol operations compared to SNMP. It is generally expected that new protocol operations will be added in the

future. This essentially means that NETCONF will likely in the future support a command-oriented approach in addition to the document-oriented approach for manipulating configuration state.

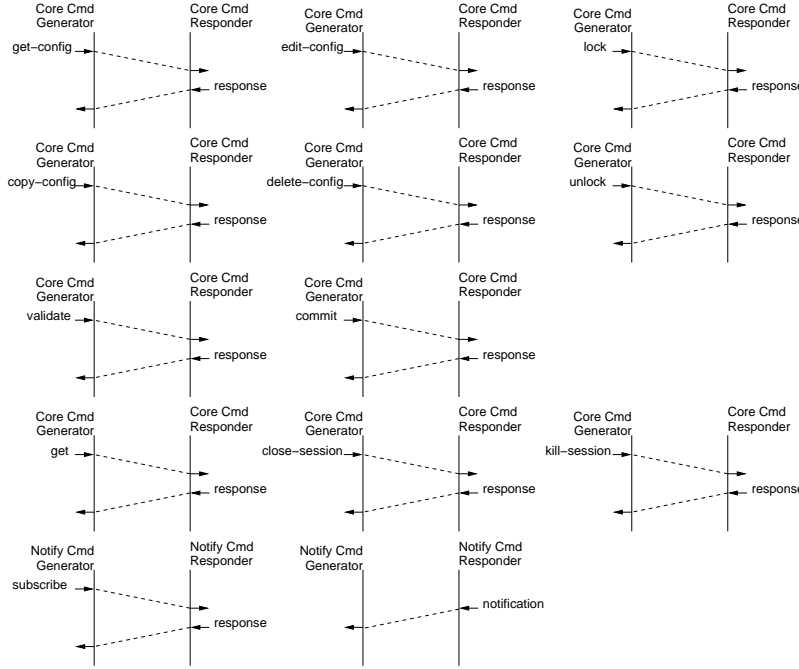


Figure 1.7: Protocol operations of NETCONF

Figure 1.7 shows the protocol operations that have been defined by the NETCONF working group. The first eight operations all operate on the configurations stored in configuration datastores. The **lock** and **unlock** operations do coarse grain locking and locks are intended to be short-lived.

The **get** operation is provided to retrieve a device's configuration state and operational state; the **get-config** operation has been provided to only retrieve configuration state. The **close-session** operation initiates a graceful close of a session while the **kill-session** operation forces to terminate a session.

The most powerful and complex operation is the **edit-config** operation. While the **edit-config** operation allows to upload a complete new configuration, it can also be used to modify the current configuration of a datastore by creating, deleting, replacing or merging configuration elements. In essence, **edit-config** works by applying a patch to a datastore in order to generate a new configuration. Since a tree-based representation, is used to express configuration state, it is necessary to describe which branches in the tree should be created, deleted, replaced, or merged. NETCONF solves this by adding so called operation attributes to an XML document that is sent as part of the **edit-config** invocation to the NETCONF server.

The `get` and `get-config` operations both support a filter parameter, which can be used to select the parts of an XML document that should be retrieved. The protocol supports a subtree filter mechanisms which selects the branches of an XML tree matching a provided template. As an optional feature, implementations can also support XPATH [14] expressions.

The notification support in NETCONF is based on an event stream abstraction. Clients who want to receive notifications have to subscribe to an event stream. An event stream is an abstraction which allows to handle multiple event sources and includes support for transparently accessing event logs. On systems which maintain an event log, it is possible to subscribe to an event stream at some time in the history and the device will then playback all recorded event notifications at the beginning of the notification stream. A subscription to an event stream establishes a filter which is applied before event notifications are sent to the client. This allows to select only the interesting messages and improves scalability.

Security

NETCONF assumes that message security services are provided by the transports. As a consequence, Figure 1.6 does not contain a security subsystem. While it has been acknowledged by the working group that an access control subsystem and model is needed, standardization work in this area has not yet been started. Some research in this area can be found in [16].

Discussion

The design of the NETCONF protocol reflects experiences made with existing proprietary configuration protocols such as Juniper's JunoScript. NETCONF addresses the requirements for configuration management protocol defined in section 1.2.1 and is therefore a good choice for this task. The pragmatic approach to layer the NETCONF protocol on top of a secure and reliable transport greatly simplifies the protocol.

The downside of NETCONF, as it is defined today, are the missing pieces, namely a lack of standards for data modeling and the lack of a standardized access control model. Since NETCONF implementations are well underway, it can be expected that these shortcomings will be addressed in the next few years and NETCONF might become a powerful tool for network configuration and trouble shooting.

1.4.3 System Logging Protocol (SYSLOG)

The SYSLOG protocol, originally developed in the 1980s as part of the Berkeley Software Distribution, is used to transmit event notification messages over the Internet. Due to its simplicity and usefulness, the SYSLOG protocol enjoys very wide deployment, even though there was never a specification of the protocol itself. This lead to a number of differences how SYSLOG messages

are handled by implementations. In 2001, an informational document was published which describes the de-facto SYSLOG protocol [33] and provides hints how implementations should deal with some of the deployed variations.

The SYSLOG protocol described in [33] has several shortcomings:

- The default transport (UDP) provides only unreliable message delivery.
- The default transport (UDP) does not protect messages. A large number of possible attacks are described in [33], among them message forgery, message modifications, and message replay.
- Some applications require mechanisms to authenticate the source of an event notification and to verify the integrity of the notification.
- SYSLOG messages contain no structured data which makes it difficult for machines to interpret them.

Since the publication of the document describing the de-facto SYSLOG protocol, the IETF has been working on a standards-track SYSLOG protocol addressing the shortcomings mentioned above [22].

Deployment Scenarios

The SYSLOG protocol can be deployed in a variety of configurations. Figure 1.8 shows some of the possible scenarios.

A SYSLOG-enabled device supports at least a SYSLOG sender. The sender can obtain event notifications from various sources. Typical sources are background processes such as routing daemons or processes implementing specific services, such as web server or database server. Another source of event notifications is often the operating system kernel, which usually provides special interfaces to pass kernel notifications to the SYSLOG process forwarding the messages.

Event notifications are collected by so called SYSLOG collectors, which typically store the notification in some database and might pass the information on to applications that are interested in processing the event notifications. Stored notifications are often used to investigate why failures happened in order to find root causes. However, stored notifications can also be useful to prove that a certain event did happen or a certain service was provided. It is therefore important that integrity, completeness, and authenticity of messages can be verified when they are fetched from the database.

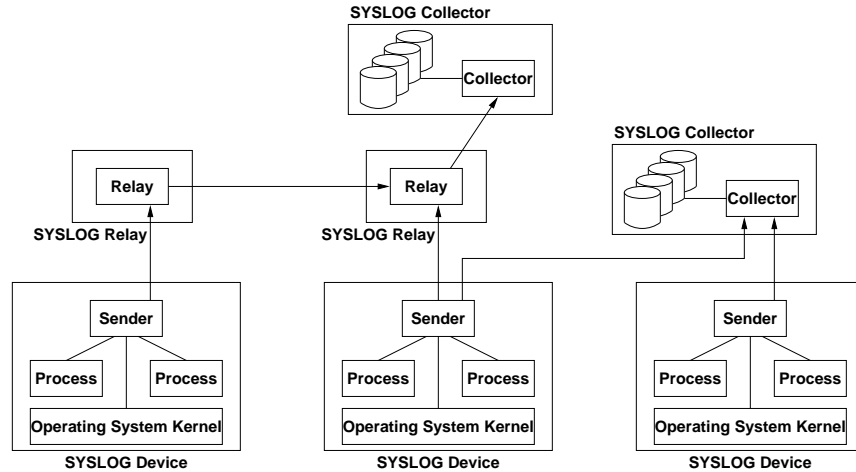


Figure 1.8: SYSLOG Deployment Scenario

The SYSLOG sender can forward event notifications via the SYSLOG protocol to notification collectors directly as shown in the right part of Figure 1.8, or it can use relays. Relays can act as simple forwarders in order to deal with firewalls or network address translators. They can, however, also be used to aggregate notifications from multiple senders or to filter notifications before passing them on towards collectors.

Architecture

The SYSLOG protocol is a so called “fire-and-forget” protocol. The originator of an event notification sends the notification to a receiver without getting an acknowledgment in any form whether the message was received, passed on to other systems, or successfully stored or processed.

SYSLOG receivers typically come in two flavors:

- A *relay* is a receiver that can receive messages and forward them to other receivers.
- A *collector* is a receiver that receives messages without relaying them.

Note that a sender is not aware whether a receiver acts as a relay or a collector. Furthermore, a single receiving entity may act as both a relay and a collector.

Figure 1.9 presents an architectural view of the SYSLOG protocol. The transport subsystem is responsible for the transmission of messages over the network. The SYSLOG protocol traditionally uses UDP as a transport. Recent work suggests to run SYSLOG over TLS [18] in order to protect messages on the wire.

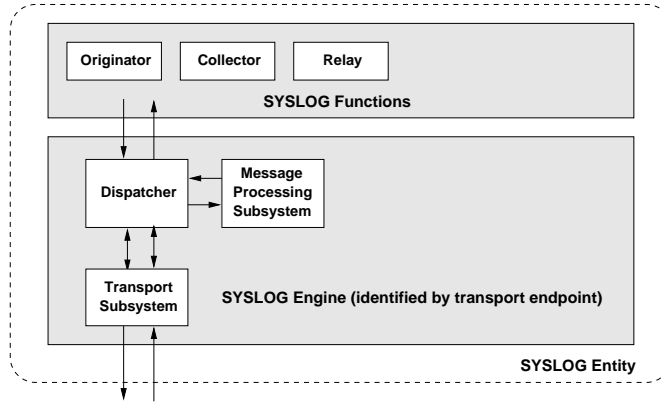


Figure 1.9: Subsystem Architecture of SYSLOG

The message processing subsystem currently includes two message processing models. The original message format is defined by the BSD SYSLOG protocol [33] while the standards-track message format is the new standards-track format developed by the IETF [22].

There are essentially three SYSLOG functions, namely an *originator* generating notifications, a *relay* forwarding notifications, and a *collector* receiving and storing notifications. As mentioned above, it is possible in the SYSLOG architecture to dispatch a received message to a relay and a collector registered on the same SYSLOG engine.

Due to the fire-and-forget nature of the SYSLOG protocol, there is no access control subsystem. An access control subsystem requires that a security subsystem establishes an authenticated identity in order to select appropriate access control rules. With a fire-and-forget protocol, it becomes difficult to establish such an authenticated identity. As a consequence, in SYSLOG there is also no need for a security subsystem.

The original SYSLOG carries only a small number of machine readable elements in notification messages, namely the identification of a facility originating the event, a severity level, a timestamp, an identification of the host originating the event, the name of the process originating the event, and optionally a process identification number. The rest of the message contains an arbitrary text message. The standards-track SYSLOG protocol essentially keeps the original format but adds more machine readable content in the form of structured data elements.

Structured data consists of zero, one, or multiple structured data elements and each structured data element contains a list of parameters in the form of name value pairs. Structured data elements are identified by a name, which is essentially a seven-bit ASCII character string. The name format supports standardized structured data element names as well as enterprise specific structured data element names. The names of the structured data element parameters

follow the same rules as structured data element names, but they are scoped by the structured data element name. The values use a simple ASCII encoding with an escape mechanism to handle special characters such as quotes.

Data Model

There is no data modeling framework yet for SYSLOG structured data elements. The specifications so far use plain text to define the format and semantics of structured data elements and their parameters.

Since SYSLOG notifications and SNMP notifications serve similar purposes, it can be expected that mappings will be defined to map SNMP varbind lists to SYSLOG structured data element parameters and also SYSLOG structured data elements into SNMP notification objects. Similar mappings already exist in many products between the more unstructured BSD SYSLOG protocol and SNMP.

Protocol Operations

The SYSLOG protocol operations are fairly simple since the protocol does not expect any bidirectional communication between SYSLOG entities.

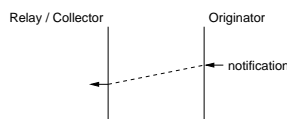


Figure 1.10: Protocol operations of SYSLOG

Figure 1.10 shows that a notification is sent to a relay or a collector. A particular feature of SYSLOG is that a SYSLOG entity can act as both a relay and a collector at the same time. This means that the dispatcher of a SYSLOG engine must be able to dispatch a received message to multiple functions, something that does not exist, for example, in the SNMP architecture (a received SNMP notification is either proxied or passed to the notification receiver).

Security

The SYSLOG protocol traditionally provides no security. Not surprisingly, this is a major issue in environments where the management network itself can't be properly secured. Furthermore, there are legislations in some countries that require operators to log certain events for later inspection and it is sometimes required to be able to prove that logged messages are unchanged and originated from the claimed source.

The SYSLOG specifications provide two mechanisms to deal with these security requirements. To provide hop-by-hop message protection, it is possible to run SYSLOG over TLS [38]. To achieve notification originator authentication

and integrity checking, it is possible to sign SYSLOG notifications [27]. This works by sending signature blocks as separate SYSLOG messages that sign the hashes of a sequence of recently sent SYSLOG notifications. Note that signature blocks can be logged together with SYSLOG messages and used later to verify the identity of the SYSLOG notification originator once the need arises.

Discussion

The SYSLOG protocol has been very successful in achieving wide spread deployment. Many C libraries support a simple API to generate SYSLOG messages and it is not unlikely that the availability of an easy to use and relatively portable API lead to wide-spread acceptance by software developers. From a conceptual point of view, SYSLOG notifications with structured data elements are close to unconfirmed SNMP notifications and it can be expected that mappings will be defined. The concept of SYSLOG relays is close to SNMP proxies, except that SNMP does not directly support a co-located relay and collector (although a fancy configuration of a proxy might achieve a similar effect).

There are some notable differences in the security services supported. SYSLOG provides a mechanism to sign notifications while SNMP provides confirmed notification delivery and access control for outgoing notifications. The SYSLOG signature mechanism might be easy to retrofit into the SNMP protocol, but confirmed notifications are more difficult to add to SYSLOG.

1.4.4 Flow Information Export Protocol (IPFIX)

In the 1990s, it became popular to collect traffic statistics about traffic flows in order to track the usage of networks. A flow is a set of IP packets passing an observation point in the network during a certain time interval. All packets belonging to a particular flow have a set of common properties [43].

Flows are a convenient mechanism to aggregate packet streams into something meaningful for the network operator. For example, a network operator might collect flow statistics for specific types of traffic in the network such as all email (SMTP) traffic, all web (HTTP) traffic, or all voice (SIP and RTP) traffic. In addition, the operator might collect flow statistics for the different customers.

In order to assign packets to flows, it is necessary to derive a set of properties. In general, the set of properties is constructed by applying a function to certain values and characteristics associated with a captured packet [49]:

1. One or more network layer packet header fields, transport layer header fields, or application layer header fields (e.g., IP destination address, destination port, and RTP header fields).
2. One or more characteristics of the packet itself (e.g., number of MPLS labels).
3. One or more fields derived from packet treatment (e.g., output interface).

Some early approaches to collect traffic statistics using the SNMP framework resulted in the Remote Monitoring (RMON) family of MIB modules [56]. While the RMON technology achieved wide deployment for collecting overall link statistics, it did not support the notion of flows very well. The Realtime Traffic Flow Measurement (RTFM) framework [3], centered around a Meter MIB module [4], details the criteria used by a meter to assign packets to flows. The RTFM framework is like the RMON framework based on SNMP technology. Even though the RTFM framework was quite flexible from the very beginning, it only achieved some moderate level of support.

In the 1990s, several vendors also started efforts to export flow information from their routers using special purpose protocols, most notably Cisco's NetFlow [9], which evolved over time to the current version 9. Cisco's NetFlow has been very successful in achieving significant deployment. This led to the formation of a working group in the IETF to standardize a flow information export protocol. The evaluation of candidate protocols [29] led to the selection of Cisco's NetFlow version 9 as a starting point for a standardization effort, which then led to the IP Flow Information Protocol (IPFIX) [49] framework.

Deployment Scenarios

The IPFIX protocol can be used in several different scenarios. Figure 1.11 shows a deployment scenario where two IPFIX devices (typically IP router) report flow statistics to a single collector. The collector stores the data in a local database. The dashed lines indicate that the collector may include another meter and exporter process to report further aggregated statistics to another collector.

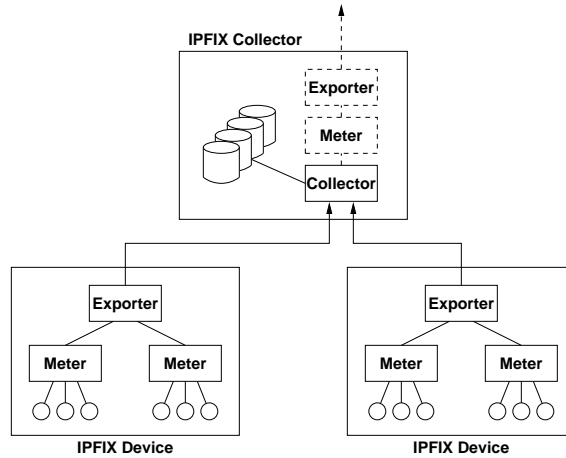


Figure 1.11: IPFIX Deployment Scenario

IPFIX devices can include several observation points (marked as circles in Figure 1.11) and meters. It is also possible to have multiple exporting processes on a single device, but this is not indicated in Figure 1.11. The internal structure

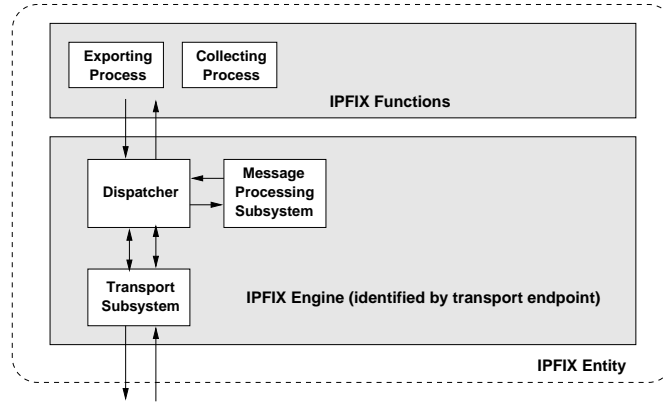


Figure 1.12: Subsystem Architecture of IPFIX

typically depends on the configuration of the device. Note that observation points and even complete meters might be running on different line cards on a high-end modular router.

Architecture

The IPFIX protocol follows a variable-oriented approach to export flow statistics [49]. Since IPFIX is a unidirectional protocol, the architecture remains fairly simple.

Figure 1.12 provides an architectural view. Like all other protocols discussed in this chapter, IPFIX has a transport subsystem which supports several different transport models. Since IPFIX is relatively new, there is currently only a single message format and message processing model. A message version number has been designed into the protocol in order to be able to deal with any changes and additions that might be necessary in the future.

There are two IPFIX functions: The exporting process is responsible for generating IPFIX messages and sending them to a collector. The collecting process receives IPFIX messages and processes them. The IPFIX requirements document also mentions concentrators capable to merge multiple IPFIX message streams and proxies, which relay IPFIX message streams [43]. None of these concepts is, however, explicitly discussed in the protocols specifications and this is why they also do not appear in Figure 1.12.

The data values transmitted in IPFIX messages belong to so called information elements. An information element is primarily identified by its element identifier, which is a numeric identifier. Globally unique element identifiers can be allocated via the Internet Assigned Numbers Authority (IANA). Vendor-specific element identifiers are created by combining the element identifier with an enterprise identifier. The later can be allocated via IANA on a first-come first-serve basis. Information elements can be scoped to a specific metering or exporting process, or they can be scoped to a specific flow measured at a specific

observation point, which again is identified by an observation identifier.

IPFIX messages can be exchanged over different transport protocols. The required to implement transport protocol is the Stream Control Transmission Protocol (SCTP) [39]. SCTP provides multiple independent streams in an association. IPFIX explores this by sending meta data and configuration data over SCTP stream zero while multiple additional streams may be allocated to send measurement data records. The optional transports for IPFIX are UDP and TCP.

Data Modeling

IPFIX uses an ad-hoc informal template mechanism for data modeling [42]. Information elements are defined by defining the following properties:

- The mandatory *name* property is a unique and meaningful name for an information element.
- The mandatory numeric *element identifier* property is used by the IPFIX protocol to identify an information element. The element identifier can be globally unique or it can be scoped by an enterprise identifier.
- The textual *description* property specifies the semantics of an information element.
- The mandatory *data type* property of an information element indicates the type of an information element and its encoding. The basic data types include signed and unsigned integers with different width (8, 16, 32, 64 bits), floating point numbers (IEEE 32-bit and 64-bit formats), a boolean type, octet string and unicode strings, several data and time types, and types for IPv4, IPv6, and MAC addresses.
- The mandatory *status* property indicates whether a definition is current, deprecated, or obsolete.
- The optional numeric *enterprise identifier* property must be present if the element identifier is not globally unique.
- Additional optional properties can indicate units, range restrictions, references and provide further type semantics (e.g., whether a numeric type is acting as a counter or an identifier).

The IPFIX specification [42] includes a non-normative XML representation of the definitions and a non-normative XML schema for the XML notation.

Protocol Operations

The IPFIX protocol is essentially a unidirectional message stream. At the time of this writing, IPFIX only has a single message format which includes a version

number, a length field, a sequence number, the observation domain identifier, and the export time [10].

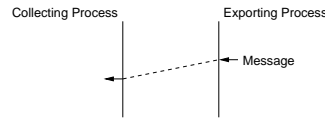


Figure 1.13: Protocol operations of IPFIX

The IPFIX message header is followed by one or more sets. Sets can be either data sets, template sets, or options template sets:

- *Template sets* are collections of one or more template records that have been grouped together in an IPFIX message. A template record defines the structure and interpretation of fields in data records. Template sets are identified by a template number which is scoped by the transport session and an observation domain.
- *Data sets* consist of one or more data records of the same type, grouped together in an IPFIX message. Data records refer to previously transported template records (using the template number). The template record defines the structure and the semantics of the values carried in a data record.
- *Options template sets* are collections of one or more option template records that have been grouped together in an IPFIX message. An option template record is a template record including definitions how to scope the applicability of the data record. Option template records provide a mechanism for an exporter to provide additional information to a collector, such as the keys identifying a flow, sampling parameters, or statistics such as the number of packets processed by the meter or the number of packets dropped due to resource constraints.

Template records establish the names and types of the values contained in subsequent data records. Compared to the name-value encoding used by SNMP and SYSLOG, IPFIX has less overhead since names are not repeated in each data record. This can significantly reduce the encoding and bandwidth overhead and makes IPFIX more efficient and scalable.

Security

The data carried by the IPFIX protocol can be used to analyze intrusion attacks or it might be the basis for billing processes. It is therefore necessary to properly maintain the integrity of IPFIX messages and to ensure that data is delivered from an authorized exporting process to an authorized collecting process. Furthermore, it is desirable to prevent disclosure of flow data since flow data can be highly sensitive.

The IPFIX protocol mandates that the Datagram Transport Layer Security (DTLS) protocol [45] is implemented for the SCTP transport. DTLS has been chosen instead of TLS for SCTP since SCTP supports unreliable and partially reliable modes in addition to the default reliable more. For the optional TCP transport, TLS must be implemented to secure IPFIX.

It should be noted that IPFIX does not have any mechanisms for access control. The configuration of an IPFIX exporter therefore implicitly implements an access control policy for flow data.

Discussion

IPFIX is derived from Cisco's NetFlow, a very successful flow information export protocol. Since IPFIX can be seen as an evolutionary improvement of NetFlow, it can be expected that IPFIX also gets widely implemented and deployed. It should, however, be noted that the requirement of the SCTP transport support raises the bar for conforming implementations. Only very recent operating systems provide full SCTP support as part of the operating system kernel.

The data modeling part of the IPFIX framework may be improved. The informal ad-hoc notation makes it difficult to build tools and it would be desirable to achieve some level of integration with the data models used by other management protocols.

1.5 Conclusions

Work on Internet network management protocols started in the late 1980s. The SNMP protocol was created at that time as a temporary solution until full OSI management protocols would take over [8]. In the mid 1990s, it became clear that this will never happen.

The Internet management protocol development and standardization was for a long time almost exclusively focused on the SNMP framework, to a large extent ignoring that many other not well standardized protocols became widely used to manage networks. In addition, management interfaces originally designed for human interaction, such as command line interfaces, became increasingly used for management automation.

It is only recently that the IETF network management community has started to realize that effective network management requires a toolbox of protocols and that a "one size fits all" approach is unlikely to succeed to address all requirements. This change of mind-set is very important, especially since many of the requirements are also much better understood today than they were some 10 years ago.

The real challenge with a multi-protocol approach is to keep the protocols aligned where necessary. Such alignment concerns mainly security related issues as it is for example operationally very important to harmonize key management procedures and access control lists. With a convergence to secure transports

and the movement to support different certificate formats by secure transports, it can be hoped that this problem will be dealt with over time.

A second area where some level alignment is very important is data modeling. It is crucial that different and potentially overlapping management interfaces operate on the same underlying data structures as this improves consistency and reduces costs. Since data modeling efforts are not as far developed as some of the protocol definitions, it can be expected that work on multi-protocol Internet management data modeling remains an active area for future research and development.

Acknowledgements

The work on this chapter was supported in part by the EC IST-EMANICS Network of Excellence (#26854).

Bibliography

- [1] U. Blumenthal and B. Wijnen. User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3). RFC 3414, Lucent Technologies, December 2002.
- [2] T. Bray, J. Paoli, and C. M. Sperberg-McQueen. Extensible Markup Language (XML) 1.0. W3C Recommendation, Textuality and Netscape, Microsoft, University of Illinois, February 1998.
- [3] N. Brownlee. Traffic Flow Measurement: Architecture. RFC 2722, The University of Auckland, GTE Laboratories, GTE Internetworking, October 1999.
- [4] N. Brownlee. Traffic Flow Measurement: Meter MIB. RFC 2720, The University of Auckland, October 1999.
- [5] J. Case, M. Fedor, M. Schoffstall, and J. Davin. A Simple Network Management Protocol. RFC 1157, SNMP Research, PSI, MIT, May 1990.
- [6] J. Case, D. Harrington, R. Presuhn, and B. Wijnen. Message Processing and Dispatching for the Simple Network Management Protocol (SNMP). RFC 3412, SNMP Research, Enterasys Networks, BMC Software, Lucent Technologies, December 2002.
- [7] J. Case, R. Mundy, D. Partain, and B. Stewart. Introduction and Applicability Statements for Internet Standard Management Framework. RFC 3410, SNMP Research, Network Associates Laboratories, Ericsson, December 2002.
- [8] V. Cerf. IAB Recommendations for the Development of Internet Network Management Standards. RFC 1052, Network Information Center, SRI International, April 1988.
- [9] B. Claise. Cisco Systems NetFlow Services Export Version 9. RFC 3954, Cisco Systems, October 2004.
- [10] B. Claise. Specification of the IPFIX Protocol for the Exchange of IP Traffic Flow Information. Internet Draft (work in progress) <draft-ietf-ipfix-protocol-24>, Cisco Systems, November 2006.

- [11] D. Clark. The Design Philosophy of the DARPA Internet Protocols. In *Proc. SIGCOMM '88 Symposium on Communications Architectures and Protocols*, August 1988.
- [12] D. Clark, L. Chapin, V. Cerf, R. Braden, and R. Hobby. Towards the Future Internet Architecture. RFC 1287, MIT, BBN, CNRI, ISI, UC Davis, December 1991.
- [13] J. Clark. RELAX NG Compact Syntax. Committee specification, OASIS, November 2002.
- [14] J. Clark and S. DeRose. XML Path Language (XPath) Version 1.0. Recommendation, W3C, November 1999.
- [15] J. Clark and M. Makoto. RELAX NG Specification. Committee specification, OASIS, December 2001.
- [16] V. Cridlig, R. State, and O. Festor. An Integrated Security Framework for XML-based Management. In *Proc. 9th IFIP/IEEE International Symposium on Integrated Network Management*, pages 587–600. IEEE, May 2005.
- [17] M. Daniele, B. Wijnen, M. Ellison, and D. Francisco. Agent Extensibility (AgentX) Protocol Version 1. RFC 2741, Digital Equipment Corporation, IBM T. J. Watson Research, Ellison Software Consulting, Cisco Systems, January 2000.
- [18] T. Dierks and E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.1. RFC 4346, Independent, RTFM, April 2006.
- [19] O. Dubuisson. *ASN.1 - Communication between Heterogeneous Systems*. Morgan Kaufmann, 2000.
- [20] R. Enns. NETCONF Configuration Protocol. Internet Draft <draft-ietf-netconf-prot-12.txt>, Juniper Networks, February 2006.
- [21] D. C. Fallside. XML Schema Part 0: Primer . W3C Recommendation, IBM, May 2001.
- [22] R. Gerhards. The syslog Protocol. Internet Draft (work in progress) <draft-ietf-syslog-protocol-17.txt>, Adiscon GmbH, June 2006.
- [23] T. Goddard. Using the Network Configuration Protocol (NETCONF) Over the Simple Object Access Protocol (SOAP). Internet Draft <draft-ietf-netconf-soap-08.txt>, ICEsoft Technologies Inc., March 2006.
- [24] D. Harrington. Transport Security Model for SNMP. Internet Draft (work in progress) <draft-ietf-isms-transport-security-model-00.txt>, Huawei Technologies, October 2006.

- [25] D. Harrington, R. Presuhn, and B. Wijnen. An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks. RFC 3411, Enterasys Networks, BMC Software, Lucent Technologies, December 2002.
- [26] D. Harrington and J. Schönwälder. Transport Subsystem for the Simple Network Management Protocol (SNMP). Internet Draft (work in progress) <draft-ietf-isms-tmsm-04.txt>, Huawei Technologies, International University Bremen, October 2006.
- [27] A. Clemm J. Kelsey, J. Callas. Signed syslog Messages. Internet Draft (work in progress) <draft-ietf-syslog-sign-19.txt>, PGP Corporation, Cisco Systems, September 2006.
- [28] E. Lear and K. Crozier. Using the NETCONF Protocol over Blocks Extensible Exchange Protocol (BEEP). Internet Draft <draft-ietf-netconf-beep-10.txt>, Cisco Systems, March 2006.
- [29] S. Leinen. Evaluation of Candidate Protocols for IP Flow Information Export (IPFIX). RFC 3955, SWITCH, October 2004.
- [30] D. Levi, P. Meyer, and B. Stewart. SNMP Applications. RFC 3413, Nortel Networks, Secure Computing Corporation, Retired, December 2002.
- [31] D. Levi and J. Schönwälder. Definitions of Managed Objects for the Delegation of Management Scripts. RFC 3165, Nortel Networks, TU Braunschweig, August 2001.
- [32] D. Levi and J. Schönwälder. Definitions of Managed Objects for Scheduling Management Operations. RFC 3231, Nortel Networks, TU Braunschweig, January 2002.
- [33] C. Lonvick. The BSD syslog Protocol. RFC 3164, Cisco Systems, August 2001.
- [34] K. McCloghrie, D. Perkins, and J. Schönwälder. Conformance Statements for SMIV2. RFC 2580, Cisco Systems, SNMPinfo, TU Braunschweig, April 1999.
- [35] K. McCloghrie, D. Perkins, and J. Schönwälder. Structure of Management Information Version 2 (SMIV2). RFC 2578, Cisco Systems, SNMPinfo, TU Braunschweig, April 1999.
- [36] K. McCloghrie, D. Perkins, and J. Schönwälder. Textual Conventions for SMIV2. RFC 2579, Cisco Systems, SNMPinfo, TU Braunschweig, April 1999.
- [37] K. McCloghrie and M. Rose. Management Information Base for Network Management of TCP/IP-based Internets: MIB-II. RFC 1213, Hughes LAN Systems, Performance Systems International, March 1991.

- [38] F. Miao and M. Yuzhi. TLS Transport Mapping for SYSLOG. Internet Draft (work in progress) <draft-ietf-syslog-transport-tls-03.txt>, Huawei Technologies, August 2006.
- [39] L. Ong and J. Yoakum. An Introduction to the Stream Control Transmission Protocol (SCTP). RFC 3286, Ciena Corporation, Nortel Networks, May 2002.
- [40] R. Presuhn. Transport Mappings for the Simple Network Management Protocol (SNMP). RFC 3417, BMC Software, December 2002.
- [41] R. Presuhn. Version 2 of the Protocol Operations for the Simple Network Management Protocol (SNMP). RFC 3416, BMC Software, December 2002.
- [42] J. Quittek, S. Bryant, B. Claise, P. Aitken, and J. Meyer. Information Model for IP Flow Information Export. Internet Draft (work in progress) <draft-ietf-ipfix-info-14>, NEC, Cisco Systems, PayPal, October 2006.
- [43] J. Quittek, T. Zseby, B. Claise, and S. Zander. Requirements for IP Flow Information Export (IPFIX). RFC 3917, NEC Europe, Fraunhofer FOKUS, Cisco Systems, Swinburne University, October 2004.
- [44] D. Raz, J. Schönwälder, and B. Sugla. An SNMP Application Level Gateway for Payload Address Translation. RFC 2962, Lucent Technologies, TU Braunschweig, ISPSOft Inc., October 2000.
- [45] E. Rescorla and N. Modadugu. Datagram Transport Layer Security. RFC 4347, RTFM, Stanford University, April 2006.
- [46] M. Rose. A Convention for Defining Traps for use with the SNMP. RFC 1215, Performance Systems International, March 1991.
- [47] M. Rose and K. McCloghrie. Structure and Identification of Management Information for TCP/IP-based Internets. RFC 1155, Performance Systems International, Hughes LAN Systems, May 1990.
- [48] M. Rose and K. McCloghrie. Concise MIB Definitions. RFC 1212, Performance Systems International, Hughes LAN Systems, March 1991.
- [49] G. Sadasivan, N. Brownlee, B. Claise, and J. Quittek. Architecture for IP Flow Information Export. Internet Draft (work in progress) <draft-ietf-ipfix-architecture-12>, Cisco Systems, CAIDA, NEC Europe, September 2006.
- [50] L. Sanchez, K. McCloghrie, and J. Saperia. Requirements for Configuration Management of IP-based Networks. RFC 3139, Megisto, Cisco, JDS Consultant, June 2001.
- [51] J. Schönwälder. Overview of the 2002 IAB Network Management Workshop. RFC 3535, International University Bremen, May 2003.

- [52] J. Schönwälder. Characterization of SNMP MIB Modules. In *Proc. 9th IFIP/IEEE International Symposium on Integrated Network Management*, pages 615–628. IEEE, May 2005.
- [53] J. Schönwälder, A. Pras, and J. P. Martin-Flatin. On the Future of Internet Management Technologies. *IEEE Communications Magazine*, 41(10):90–97, October 2003.
- [54] D. Steedman. *Abstract Syntax Notation One (ASN.1): The Tutorial and Reference*. Technology Appraisals, 1990.
- [55] F. Strauß and J. Schönwälder. SMIng - Next Generation Structure of Management Information. RFC 3780, TU Braunschweig, IU Bremen, May 2004.
- [56] S. Waldbusser, R. Cole, C. Kalbfleisch, and D. Romascanu. Introduction to the Remote Monitoring (RMON) Family of MIB Modules. RFC 3577, AT&T, Verio, Avaya, August 2003.
- [57] M. Wasserman and T. Goddard. Using the NETCONF Configuration Protocol over Secure Shell (SSH). Internet Draft <draft-ietf-netconf-ssh-06.txt>, ThingMagic, IceSoft Technologies, March 2006.
- [58] B. Wijnen, R. Presuhn, and K. McCloghrie. View-based Access Control Model (VACM) for the Simple Network Management Protocol (SNMP). RFC 3415, Lucent technology, BMC Software, Cisco Systems, December 2002.
- [59] T. Ylonen and C. Lonvick. The Secure Shell (SSH) Protocol Architecture. RFC 4251, SSH Communications Security Corp, Cisco Systems, January 2006.

Management of Ad-Hoc Networks

Remi Badonnel, Radu State, Olivier Festor

This chapter presents a state of the art of the management of ad-hoc networks and is structured as follows. We introduce ad-hoc networking and the challenges of ad-hoc networking management in Section 1. We present the main management paradigms and the underlying architectures in Section 2. Several application domains including monitoring, configuration and cooperation control are covered by Section 3. Finally, Section 4 concludes the chapter with a synthesis of research orientations in this domain.

1 Introduction

The increasing need of mobility in networks and services leads to new networking paradigms including ad-hoc networking [27, 39, 31] where the mobile devices can establish direct and multi-hop communications among them.

Definition 1 *A mobile ad-hoc network (MANET) is a self-organizing network that can be spontaneously deployed by a set of mobile devices (laptops, phones, personal assistants) without requiring a fixed network infrastructure.*

The deployment of an ad-hoc network is performed dynamically by the mobile devices (or nodes) themselves: each mobile node is both a terminal that communicate with the others and a router that can forward packets on behalf of the other nodes. If two network nodes cannot communicate directly because they are not localized in the same one-hop neighborhood, a multi-hop communication is established using intermediate nodes. Ad-hoc networks provide localized and temporal-limited connectivity in dedicated target environments: search and rescue services, intervehicular communications, last mile Internet delivery and geographically challenging environments. The emergence of new applications is favored by the advances in hardware technologies and by the standardization of ad-hoc routing protocols.

Compared to fixed networks, ad-hoc networks permit to reduce the costs of deploying and maintaining the infrastructures but they present other additional constraints such as:

- dynamic topology: such networks do not require a fixed infrastructure. They are spontaneously deployed from mobile nodes and are then highly dynamic, since nodes might come and go based on user mobility, out-of-reach conditions and energy exhaustion.

- heterogeneity of devices: a typical ad-hoc network is composed of various devices such as laptops, personal digital assistants, phones and intelligent sensors. These devices do not provide the same hardware and software capabilities but they have to interoperate in order to establish a common network.
- distributed environment: the network running is based on the cooperation of independent nodes distributed over a given geographical area. The network maintenance tasks are shared among these nodes in a distributed manner based on synchronization and cooperation mechanisms.
- resource constraints: bandwidth and energy are scarce resources in ad-hoc networks. The wireless channel offers a limited bandwidth, which must be shared among the network nodes. The mobile devices are strongly dependent on the lifetime of their battery.
- limited security: ad-hoc networks are more vulnerable than fixed networks because of the nature of the wireless medium and the lack of central coordination. Wireless transmissions can be easily captured by an intruding node. A misbehaving node can perform a denial of service attack by consuming the bandwidth resources and making them unavailable to the other network nodes.

The key concept of ad-hoc networking is the routing mechanism. Ad-hoc networks rely on a routing plane capable to fit with their dynamics and resource constraints. In particular, an ad-hoc routing protocol must be capable to maintain an updated view of the dynamic network topology while minimizing the routing traffic. Numerous ad-hoc routing protocols were proposed in the last ten years. The MANET working group [22] manages the standardization of ad-hoc IP routing protocols at the IETF and classifies them in two major routing approaches: reactive routing and proactive routing.

Definition 2 *A reactive routing protocol is a routing protocol which establishes the routes in an on demand manner only when a route is requested by a node, by initiating a route discovery process.*

In the reactive approach, a source node had to flood a request for a route to a destination node in the ad-hoc network. The intermediate nodes are discovered and memorized during the flooding. When the destination node receives the request, it uses this reverse path to contact the source node and to provide the requested route. A typical example of reactive protocol is the Ad-hoc On-demand Distance Vector (AODV) protocol described in [30].

Definition 3 *A proactive routing protocol is a routing protocol which periodically maintains the network topology information on each network node, by managing local routing tables.*

In the proactive approach, each node maintains a routing table that is regularly updated by exchanging topology information with the other nodes. An

example of a proactive protocol is the Optimized Link State routing protocol (OLSR) [15]. Each node performs two main operations: it determines the list of direct-connected neighbor nodes by accomplishing link sensing through periodic emission of beaconing hello messages and exchanges link state information with the other nodes by flooding topology control messages. The OLSR heuristic reduces the control overhead by selecting a subset of nodes called Multi-Point Relays (MPRs) responsible for forwarding broadcast messages during the flooding process.

The two routing approaches do not present the same properties and the choice of the one or the other is done according to the particular application. The reactive routing approach reduces significantly the control overhead since it does not require to flood topology information periodically. However, the proactive routing approach offers a lower latency as the route discovery has not to be done. Hybrid routing protocols provide a trade-off between control overhead and latency.

While the routing plane is progressively converging towards the use of standardized protocols, the management plane raises interesting research challenges towards providing a management framework to ad-hoc networks.

1.1 Management Requirements

A fixed network management model is not directly appropriate for ad-hoc networks. New management paradigms and architectures must be defined to cope with the dynamics and resource constraints of such networks. Managing ad-hoc networks presents us with different research issues which are not encountered in the same way in the common fixed networks:

- management domain related: since ad-hoc networks are dynamically formed by loosely coupled entities, the notion of administrative domain, responsible for the overall management is difficult to define. Although, from a technical point of view, cooperative and role-based schemes are possible such that the manager/agent roles are assigned dynamically, the management layer has to implement higher level business policies, which are hard to define in blurry shaped administrative domain.
- relevant management information: what should be monitored and what actions to be taken is still unknown in this case. The challenge is to identify the essential pieces of information relevant in this context and to determine the possible corrective configuration operations in a close-loop management paradigm.
- management agent reliability and willingness to cooperate: the management agent is constrained by its inherent limits (connectivity, battery, failures) so that the provided monitored data might be biased. Additionally, malicious or non-cooperative nodes might provide false data or no data at all. The key issue is to define management paradigms, where biased or faulty information can be out-weighted or discarded as such.

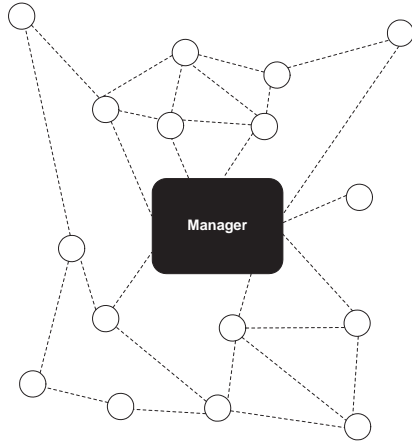
- cost of management: resources (essentially network bandwidth and battery power) are already scarce in ad-hoc networks and monitoring and management actions are additional consumers in an already resource-limited context. Defining lightweight schemes, where already available information is used and resource availability and consumption can be minimal, are the main issues.
- self-organization: the dynamic nature of mobile ad-hoc networks requires that the management plane is able to adapt itself to the heterogeneous capacity of devices and to the environment modifications. The ad-hoc network is deployed in a spontaneous manner, the management architecture should be deployed in the same manner in order to minimize any human intervention.
- robustness and scalability: the management plane must be easy to maintain and must remain coherent, even when the ad-hoc network size is growing, when the network is merging with an other one or when it is splitting into different ones.

New management paradigms were considered to deal with the research issues posed by ad-hoc networks. They also had to be integrated into the common management approaches of fixed networks.

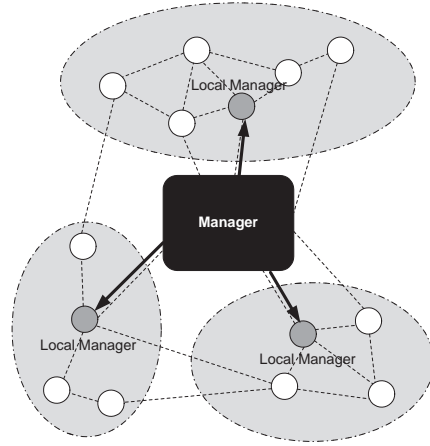
1.2 Management Organizational Models

An ad-hoc network corresponds to a distributed environment where different organizational models can be considered for the management purposes. A management organizational model defines the role (manager or agent) of each component of the management plane and specifies how each component interacts with the others. Typically, we can distinguish four different management organizational models presented on figure 1:

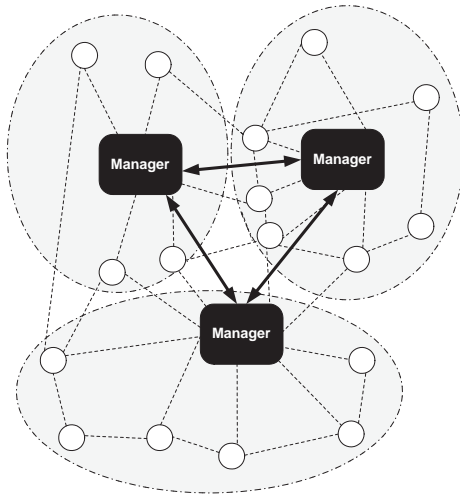
- centralized model (figure 1(a)): a single network manager manages all the nodes in the whole network. The significant advantage of this model is to simplify the management task by considering a single point of control. However, the centralized model can generate a high management traffic and is not robust since the failure of the manager leads to the failure of the whole management system.
- centralized hierarchical model (figure 1(b)): the network manager uses local managers as intermediary to perform some management tasks. Each local manager manages a subset of network nodes and is assigned with a given degree of responsibility. The manager is the central point of control since it has the highest degree of responsibility.
- distributed model (figure 1(c)): the management is performed by a set of managers communicating according to a peer-to-peer scheme. Each manager has the same degree of responsibility and is fully responsible for



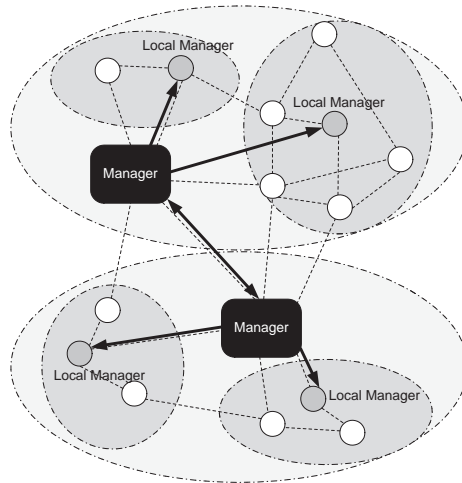
(a) Centralized management



(b) Centralized hierarchical management



(c) Distributed management



(d) Distributed hierarchical management

Figure 1: Organizational models for ad-hoc network management

managing a subset of nodes in the network. The usage of multiple managers reduces the overhead of management messages and improves the robustness of the management system. However, a cooperative mechanism must be defined in order to ensure the coherence of the management operations taken by managers.

- distributed hierarchical model (figure 1(d)): this model is a distributed model where each of the managers can delegate part of its management tasks to local managers.

These organizational models are not specific to ad-hoc networking but are recurrent in network management. Because of the nature of ad-hoc networks, the role assignation and the relationships among components are highly dynamic compared to regular fixed networks.

2 Management Paradigms

This section presents the main management paradigms for mobile ad-hoc networks and describes the underlying architectures able to cope with the constraints of such dynamic and resource scarce networks.

2.1 SNMP-based Management

Current network management systems use a set of available management protocols (CLI [6], SNMP [38], XML-based [18]) to exchange management information between a network manager and the agents. The Simple Network Management Protocol (SNMP) [38] is among the most common used protocol for network monitoring and configuration. Integrating ad-hoc networks into common agent/manager models require the compatibility with this protocol, but also requires in the same time a reorganization of the management plane in order to limit the underlying network overhead.

2.1.1 ANMP approach

Among the pioneering management approaches for ad-hoc networks, the Ad-Hoc Network Management Protocol (ANMP) [14] architecture defines a centralized hierarchical solution based on the SNMP protocol. The compatibility relies on using the same protocol data unit structure and the same management information base structure as in SNMP. The major difference comes from the architectural model (presented in figure 2), which combines clusters and hierarchical levels. ANMP comes within the scope of management by delegation. It clusters the management plane and organizes it in a three-level hierarchy composed of: individual managed nodes interacting as agents at the lowest level, clusterheads interacting as local managers at the middle level and a central network manager at the top level.

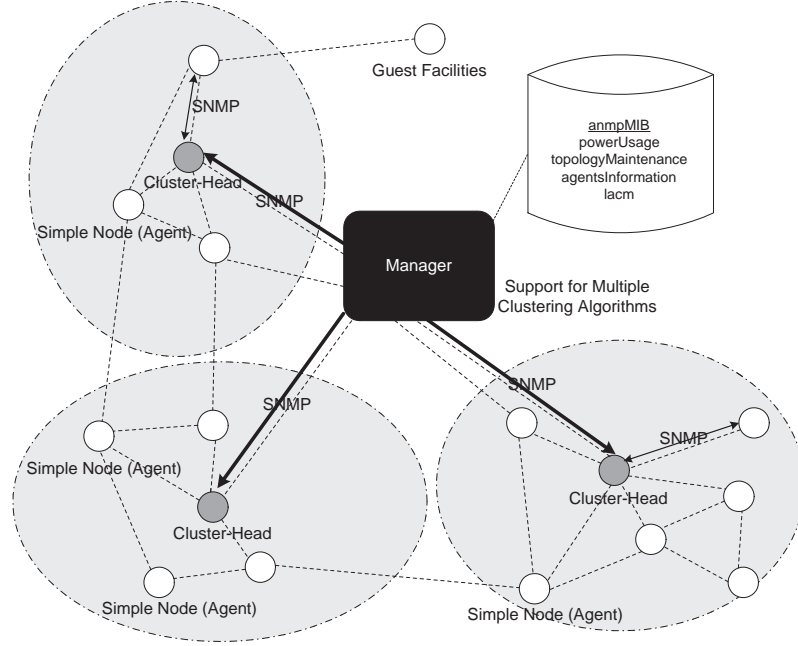


Figure 2: ANMP architecture

The hierarchical model reduces the network traffic overhead by collating management data at the intermediate levels, but is limited to three hierarchical levels in order to facilitate the maintenance of the hierarchy in such dynamic networks. Two clustering algorithms are defined at the application layer to organize the management plane:

- The first approach is based on graph-based clustering and forms clusters of one-hop neighbor nodes based on the network topology. In the network topology seen as a graph, each mobile device is represented by a node identified by a unique ID (MAC address, serial number) and a bidirectional link exists between two nodes if and only if they are within the transmission range of each other.

The algorithm performs in a distributed manner the organization of one-hop clusters where the node with the minimum ID among its neighbors is elected as the clusterhead. Each node implements internal data structures including the list of one-hop neighbors, the list of nodes in the cluster and a ping counter indicating last time a ping was received from the clusterhead.

- The second approach called geographical clustering forms clusters of up to three-hop neighbors based on the spatial density. The approach requires that node positions are known through a geographical positioning system

such as GPS. It is based on two algorithms: a centralized periodic processing for cluster definition and a distributed maintenance processing to manage node mobility.

The centralized processing considers the ad-hoc network as a rectangular box and calculates horizontal and vertical stripes in order to split the network into rectangular clusters of different sizes (up to three times the transmission range) but of homogeneous spatial density. In each of these rectangular clusters, the ad-hoc node located in the box center is defined as the clusterhead.

The clustering processing is executed periodically and a maintenance processing manages node mobility between two clustering updates. The maintenance processing performs at the cluster level defines an announcement protocol for nodes to leave and join a cluster and provides guest facility in each cluster for nearby unmanaged nodes.

ANMP is not limited to the two presented approaches but can be extended to other clustering algorithms. In particular, while ANMP makes a distinction between clustering at the application layer for management purposes and clustering at the routing layer, it may be suitable to propose a lightweight management approach by piggybacking an ad-hoc network routing protocol.

A management information base (MIB) is defined by ANMP as an additional group to the standardized MIB-II of SNMP. This group called *anmpMIB* includes security information data to implement the Level-based Access Control Model (LACM) in order to secure management operations and management information access. In the management plane, each node has a clearance level: the higher the level is, the lower the security level is. The group also defines topology information including two information subgroups for each clustering approach, and agent information statistics maintained by managers. Finally, the group describes objects related to device battery and power consumption, including a drain function of the battery used for predictive purposes.

ANMP defines a complete SNMP-compatible management approach including a security model. The clustering schemes can ensure that over 90 percents of nodes are managed while keeping a low network management traffic overhead. ANMP introduces a first identification of a need for some active code facility by considering a programmable framework to extend the functionalities of agents.

2.1.2 GUERRILLA approach

An alternative solution is proposed in [37] by the GUERRILLA management architecture, which relies on two concepts: management plane self-organization and dynamic deployment of management related code. GUERRILLA proposes a context-aware approach where management intelligence is spread over the network nodes according to their capabilities. It aims at making management operational in highly dynamic networks by maintaining connectivity in the management plane (and enabling disconnected management) and by minimizing management costs on node resources.

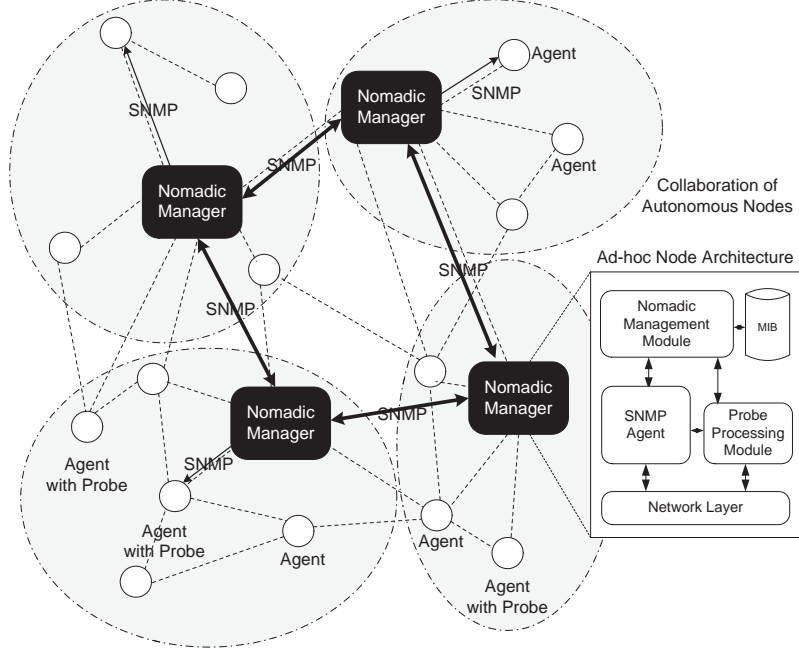


Figure 3: GUERRILLA architecture

This distributed architecture (presented in figure 3) is composed of nomadic managers (higher tier) and active probes (lower tier):

- A nomadic manager maintains the connectivity in the management plane with the other managers and updates the management plane configuration in an autonomous way.
- Deployed by a nomadic manager, an active probe is an extension to an SNMP agent. It integrates an execution environment for hosting monitoring scripts and performs the localized management operations.

The clustering algorithm relies on the Cluster-based Topology Control (CLTC) approach which minimizes the consumption of the node resources. Moreover, the nomadic managers implement an auto-configuration mechanism to determine the management actions to be executed in an adaptive manner. In a formal way, a nomadic manager determines the management action a – from a set of possible actions A – that maximizes the objective function $f(a)$ defined by equation 1.

$$f(a) = \frac{U(T(s, a)) - U(s)}{C(a)} \quad (1)$$

In this equation, s represents the view of the domain state perceived by the manager and $T(s, a)$ is a transition function that provides the new state of the

domain predicted by the manager if the action a is executed. $U(s)$ defines a utility function possibly derived from policies, while $C(a)$ represents the management cost function. By maximizing the function $f(a)$, the nomadic manager aims at improving the managed environment utility with a minimal cost.

GUERRILLA comes within the scope of management by delegation. It proposes fundamental good assumptions such as self-organization, disconnected management and takes into account heterogeneous devices.

2.2 Policy-based Management

Policy-based management (PBM) consists in defining high-level management objectives based on a set of policies that are enforced in the network. Architectures for policy-based management rely on two main functional components:

- the Policy Decision Point (PDP) component which aims at converting the high-level policies into low-level device-dependent policies and performing their distribution,
- the Policy Enforcement Point (PEP) component which enforces the low-level policies they receive from the PDP component.

The Common Open Policy Service (COPS) protocol was specified to standardize the distribution policy through the PDP-PEP communications. Several research work in [26, 32, 13] focus on the adaptation of the policy-based scheme to ad-hoc networking constraints.

2.2.1 Decentralized COPS-PR approach

A first management framework is introduced by Phanse in [32, 33] to extend the policy-based management for quality of service in ad-hoc networks. Phanse decentralizes COPS for provisioning (COPS-PR). The latter is an extended protocol which combines the outsourcing and provisioning models allowing to support hybrid architectures.

This distributed hierarchical approach is described on figure 4. It relies on three key mechanisms :

- clustering and cluster management: the clustering of the ad-hoc network relies on a simple k-hop clustering scheme: during the initial deployment, some Policy Decision Point nodes called super-PDPs are present in the network and form clusters with the PEP client nodes located in the k-hop neighborhood. The number of intermediate nodes between a super-PDP server and its PEP client nodes is therefore bounded to k-hops and the management system can expect to provide acceptable response time performance.
- dynamic service redundancy: a mechanism for dynamic service redundancy is used when the super-PDP servers are not enough numbered to serve all the clients in the network, or when clients are moving from one

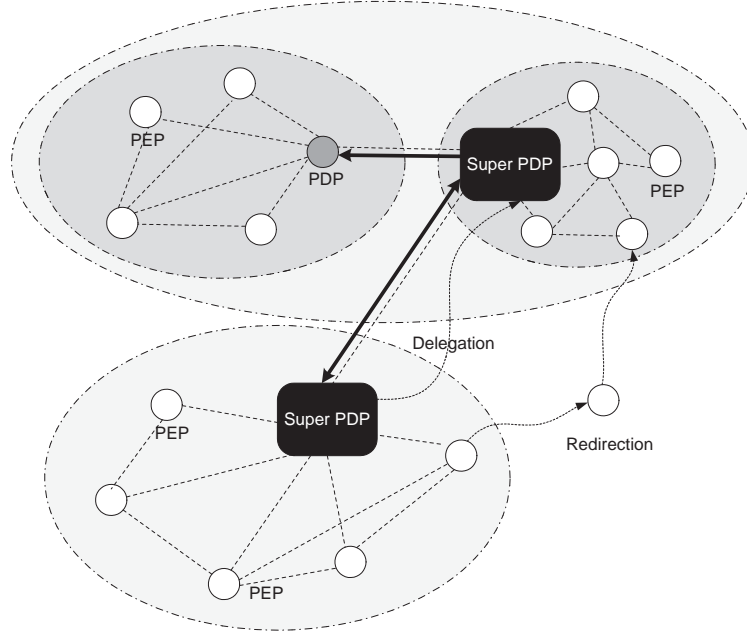


Figure 4: Policy-based management with decentralized COPS-PR

cluster to another one. It consists of a delegation service and a redirection service. The delegation service answers the lack of super-PDP servers in the network, by allowing a super-PDP server to delegate part of its decisions to another node interacting as a regular PDP server. In this case, the super-PDP server keeps a hierarchical control on the regular PDP servers. The redirection service manages the PEP clients moving among network clusters by allowing a PDP server to dynamically redirect a client to a more appropriate less distant PDP server.

- service location discovery: a discovery mechanism is proposed to automatically discover new policy servers but it is not yet designed.

The management framework supports multi-domain policy negotiation for PEP client visitors from other domains. An inter-PDP server signaling is introduced in COPS messages to exchange policies among PDP servers from different domains. If the visited PDP server cannot find policies for a visitor PEP client, it queries the PEP client to get its home PDP server address and negotiates the policies with the home PDP server.

Phanse's approach is an innovating solution for policy management in ad-hoc networks, which was prototyped and validated experimentally in a multi-hop environment. However, the considered communication protocol COPS-PR is not

widely used in today's network management systems. Some improvements could be possible for the multi-domain policy negotiation mechanism. In particular, the policy could be transferred during the redirection and the PEP clients could come with their own policies.

2.2.2 DRAMA approach

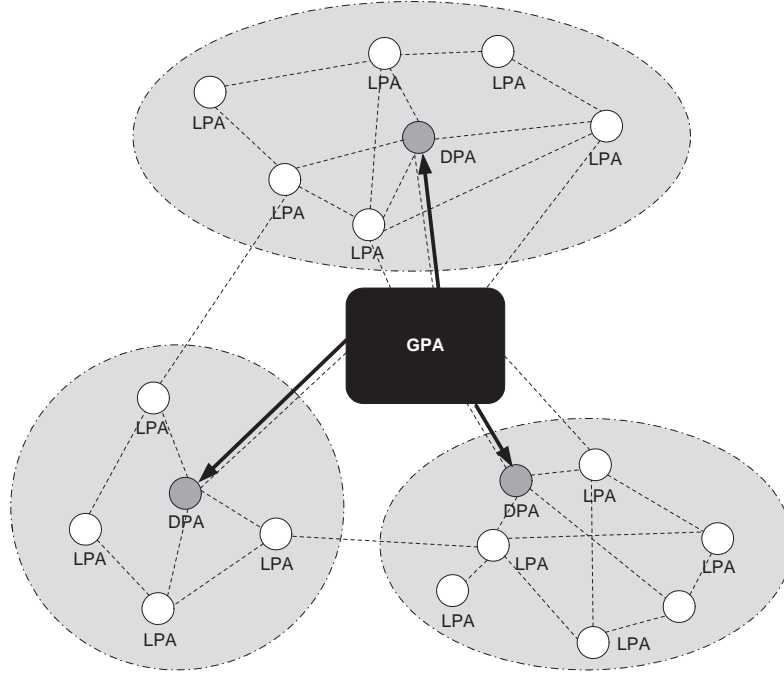


Figure 5: DRAMA architecture

DRAMA defines a centralized hierarchical policy-based management system for ad-hoc networks in [13, 12]. This three-level architecture described on figure 5 is composed of a set of policy agents having the same basic structure. The Global Policy Agent (GPA) at the top level manages multiple Domain Policy Agents (DPA). A Domain Policy Agent manages in turn multiple Local Policy Agents (LPA). Network management policies are propagated hierarchically from the GPA agent to the DPAs and from the DPAs to the LPAs.

The conceptual architecture is relatively simple, but allows to perform an exhaustive set of experiments: the management system was prototyped and demonstrated in a realistic environment to illustrate several use cases including CPU utilization reporting, server reallocation upon failure, reconfiguration of bandwidth allocation.

2.2.3 Node-to-node approach

A node-to-node approach is defined in [26] in the framework of corporate ad-hoc networks presenting uniform management and company-wide policies.

Definition 4 *The node-to-node policy distribution defines a fully distributed policy distribution where a network node exchanges and synchronizes the policies with the neighboring nodes (see figure 6).*

This distributed architecture works over the DiffServ quality-of-service model providing the network layer differentiated service: the communications among network nodes can be marked with different priority levels.

In this architecture, each ad-hoc device exports both DiffServ functions: the edge router monitors and marks local application traffic and the core router classifies and schedules routed packets. From the management plane point of view, the ad-hoc devices embed both PDP and PEP functionalities including a local policy repository to store the policy rules applicable to the device.

The key concept of the architecture is based on a two-phase policy distribution using a synchronization mechanism:

- In the first phase of the distribution, the ad-hoc device of the administrator interacts as a Policy Decision Point (PDP) and distributes the policies to the other devices behaving as Policy Enforcement Point (PEP) and storing the policies on their local repository. This phase is not sufficient to configure all the ad-hoc nodes if these were not connected to the network at the distribution time.
- To propagate the policies to all the devices, the second phase consists in a permanent synchronization among two neighboring nodes. An already-configured node can interact as a local PDP to configure a device which has not received the latest policies.

An authority mechanism allows a management hierarchy with delegation. Each manager has an authority level known by all. Network policies are tagged with manager identifiers. The more authoritative policies always replace the less authoritative ones. The approach describes an interesting device-level management framework for enterprise networks. It assumes that in the future quality-of-service mechanisms will be available at the MAC layer. However, the synchronization protocol and the consistency problems of the policies are not discussed.

2.2.4 Scalability evaluation

Burgess evaluates the scalability of policy-based configuration management for ad-hoc networks in [9]. The performance evaluation relies on an analytical framework which takes into account the unreliable nature of the communication medium and the mobility factors in ad-hoc networks.

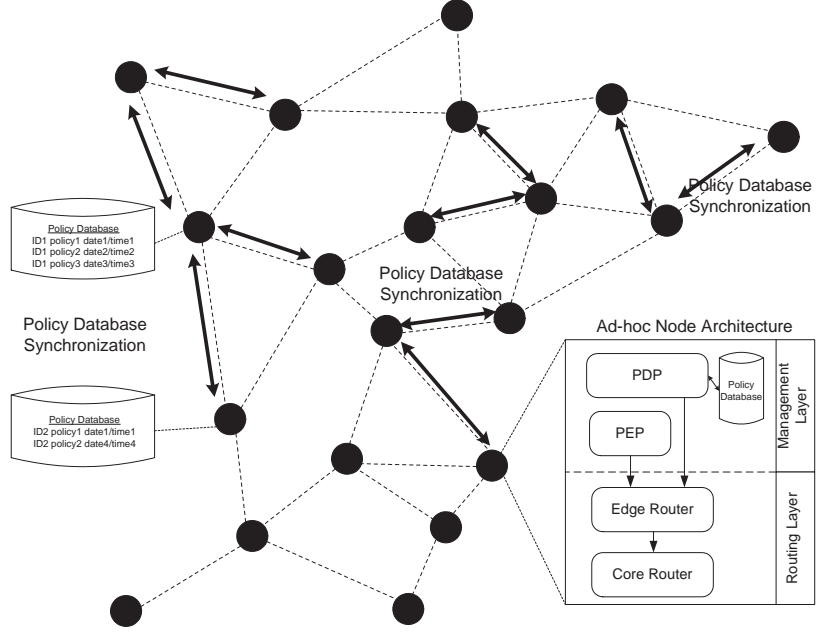


Figure 6: Node-to-node policy-based management

Hypothesis 1 *Formally, an ad-hoc network can be represented by a connectivity matrix A where rows/columns stand for an ad-hoc node. A matrix entry $A[i, j]$ indicates whether node i is connected to node j . The connectivity among two peers of nodes undergoes time and mobility caused variations in the context of ad-hoc networks. The matrix entry value can be therefore defined by a probability value.*

The connectivity matrix serves as an analytical basis for analyzing the network traffic overhead generated by different policy-based management models and for evaluating their scaling behavior.

The organizational models presented in Section 1 are refined for the context of policy-based management and leads to a set of six different management models. Each of these models corresponds to a different degree of decentralization with respect to three main criteria:

- policy decision: the deployment of policies can be controlled by a central manager or by a set of distributed managers. The final hosts can either strictly respect these policies or can freely modify some aspects of them.
- policy exchange: the policies can be received from a manager, from an intermediate manager in a hierarchical manner, or from another node in a node-to-node scheme,

- policy enforcement: the policies can be transmitted by a manager or can be locally enforced by the hosts.

The scalability analysis relies on a key assumption: the probability of a successful configuration depends on the regularity of the maintenance operations, which in turn depends on the reliability of the communications between the policy source and the hosts. In order to estimate the efficiency of the configuration, the average rate of errors generated by an host is compared to the average rate of repair maintenance operations which can be done in function of the communication channel size and reliability. The configuration is successful if and only if all the host errors have been repaired. Therefore, the average rate of possible repair operations must be higher than the average rate of host errors.

The results of this analysis provide a stochastic quantification of the scalability of policy-based management schemes. They confirm the limits of a centralized management due to the communication bottleneck caused by the centralized manager. Distributed management schemes offers a better robustness and scalability but introduce policy convergence issues.

2.3 Middleware for Management

An ad-hoc network is an heterogeneous set of nodes with diverse capabilities in terms of energy and processing facilities but also in terms of operational protocols and services. A programmable middleware defined by [20] provides a common communication infrastructure capable to align dynamically the software capabilities of network nodes. The middleware defines a programmable scheme where each node is able to share, download and activate loadable plugins offering specific operational functionalities such as an application service or a routing protocol. The key idea is to elect a common set of loadable plugins and to distribute it among the network nodes in order to facilitate their communications.

The middleware for management is organized into clusters, where the cluster head coordinates the election and distribution of loadable plugins. In a given cluster, each node exposes several loadable plugins and stores locally the executable code in its plugin repository. The cluster head initiates the election process by contacting the cluster nodes and requesting them to advertise the candidate plugins. Based on the set of advertised plugins P , the cluster head uses an election algorithm which determines the most appropriate(s) plugin(s) for the given context. The selection of plugins aims at selecting the plugins which minimize the cost function $g(p)$ defined in equation 2.

$$g(p) = \sum_{i=1}^n w_i \times C_i(p) \quad (2)$$

where $w_i \in [0, 1]$, $\sum_{i=1}^n w_i = 1$ and $C_i \in [0, 10]$, $\forall i \in [1, n]$

The characteristics $C_i(p)$ of a given plugin p are rated from 0 to 10 and are assigned with a mathematical weight w_i by the cluster head. This cost function considers the characteristics such as CPU usage, memory consumption and plugin size to converge to a common basis of plugins among the cluster nodes. The

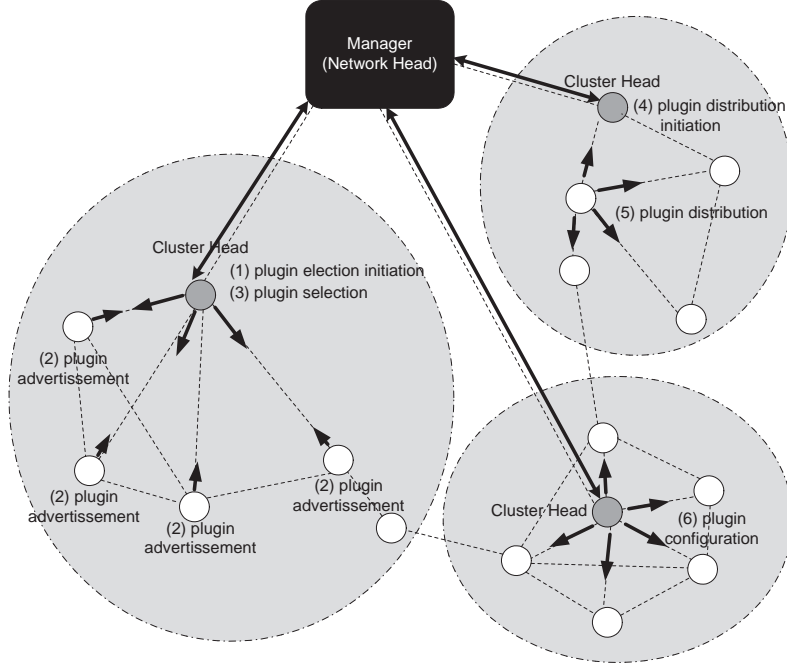


Figure 7: Middleware for management

cluster head initiates then the plugin flooding by requesting the cluster nodes that own the plugins to distribute them to their one-hop neighbors in the cluster in a peer-to-peer manner. Each cluster mate stores the common plugins into the plugin repository and activates them. The cluster head is capable to configure and to modify on-the-fly the parameters of the current activated plugins.

The middleware infrastructure was implemented by combining a lightweight XML-based message-oriented protocol and the Java platform for small and medium devices. The first evaluation performance shows the management middleware is efficient in terms of convergence and scalability. The middleware was limited to a centralized scenario with a single cluster and must be extended to multiple clusters with cluster-to-cluster communications.

2.4 Probabilistic Management

A pure and sound management approach, where all nodes are managed at any time is too strict for ad-hoc networks.

Definition 5 *Instead of addressing the management of the whole network, a probabilistic management approach focus on a subset of network devices or on a subset of network parameters to derive the behavior of the whole network with a given probability guarantee.*

A probabilistic management approach is proposed in [3] for ad-hoc networks based on probabilistic guarantees. A subset of nodes is considered to be managed in order to provide a light-weight and reliable management. These nodes are determined based on their network behavior so as to favor subsets of well-connected and network participating nodes. The assumption of the probabilistic management approach was motivated by the simple observation that ad-hoc network management should not comprise the totality of network elements (as it is typically the case in fixed networks).

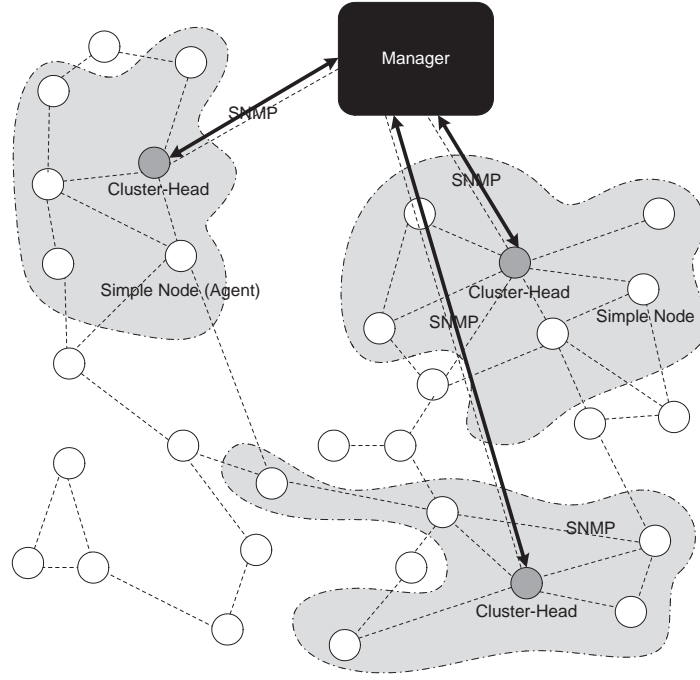


Figure 8: Probabilistic management

The requirements on the management plane are relaxed and are limited to a subset of nodes called a spatio-temporal connected component. This component is defined as a subset of the ad-hoc network, such that nodes within such a component have a high probability of being directly connected. The term spatial derives from this neighborhood notion, while the term temporal is related to the temporal behavior of this neighborhood. In a store and forward oriented architecture, such nodes are also capable to inter-communicate at a higher hop-count. The probabilistic management is limited to the first and eventually second largest spatio-temporal connected component. With respect to such a selective management scheme, probabilistic guarantees are determined in function of the percentage of nodes to be managed. The underlying algorithm describes the extraction of spatio-temporal components from a probability

matrix similar to the matrix A defined in Section 2.2.4. The election of manager nodes in each component can be performed with a centrality measure: the most central nodes of a component are elected as managers.

Definition 6 *The centrality is a measure capable to quantify the relative importance of a node in a network or a graph structure. Several variants of this measure have been defined in social networking [7].*

A spatio-temporal component is a non-oriented connected graph associated to a m -dimensional matrix M . A typical centrality measure that can be applied on this graph is the degree centrality c_{degree} defined by equation 3. A node i is considered as central if it is directly connected to a maximal number of other nodes.

$$c_{degree}(i) = \sum_{j=1}^{j=m} M(i, j) \text{ for a given node } i \quad (3)$$

The degree centrality is a local measure limited to the one-hop neighborhood information. An alternative centrality measure is the eigenvector centrality defined by Bonacich in [4]. The key idea is to define the centrality measure in a recursive way: the more a node is directly connected to central nodes, the more the node is central. For such a non-oriented connected graph, Bonacich demonstrates that the problem can be reduced to the typical eigenvector problem defined by equation 4.

$$M \times \vec{x} = \lambda \times \vec{x} \quad (4)$$

The vectors that solve this equation are called eigenvectors and are paired with corresponding eigenvalues. The principal eigenvector $\vec{v}_{principal}$ is the solution vector which is paired with the highest eigenvalue. The eigenvector centrality $c_{eigenvector}$ of node i corresponds to the i -th element of the principal eigenvector of matrix M , as mentioned in equation 5.

$$c_{eigenvector}(i) = \vec{v}_{principal}(i) \text{ for a given node } i \quad (5)$$

A generalized version of the eigenvector centrality has been defined in [5] in the case of oriented connected graphs.

Another probabilistic scheme is designed at the level of the information model in [16]. It aims at improving the acquisition of monitoring information in an ad-hoc network. In this scheme, monitoring information is not directly queried from the network devices but must be processed by a correlation-based probabilistic model. The direct access to raw monitoring data is not possible. An information query must be formulated to the acquisition module which integrates the probabilistic model. This model relies on time-varying multivariate normal distributions and allows to correlate multiple raw monitoring data through statistical techniques in order to improve the information reliability. Disparate and faulty monitoring raw data may generate biased approximated information. The correlation mechanism thus provides information approximations with probabilistic confidences.

The scheme allows to reduce the number of nodes involved in the data monitoring according to the asked confidence. The architecture defines a novel approach to integrate an information base with a correlation probabilistic model and provides a trade-off between information confidence and data acquisition cost. However, the efficiency of the architecture is directly correlated to the availability of efficient probabilistic models for a given monitored metric.

3 Application Domains

This section details the research work addressing management applications in MANETs including network monitoring, configuration of IP addresses and control of node cooperation.

3.1 Monitoring

Monitoring ad-hoc networks resides in measuring performance metrics, collecting the measurement data and processing these measurements.

3.1.1 WANMON approach

A local monitoring approach called WANMon (Wireless Ad-Hoc Network Monitoring Tool) is proposed in [29] for determining the resources usage of an ad-hoc node. The architecture is locally deployed such that an agent is responsible for monitoring the resources of an ad-hoc device. It monitors the resource consumption generated by ad-hoc network-based activities: network usage, power usage, memory usage and CPU usage. In particular, the tool aims at distinguishing both the resources consumed for the node itself and the resources consumed by the routing activity on the behalf of other nodes. WANMon is one of the first monitoring tools dedicated to ad-hoc networking, but presents some limitations. The tool cannot provide real-time information and is limited to a local monitoring activity.

3.1.2 DAMON approach

DAMON described in [34] defines a Distributed Architecture for MONitoring multi-hop mobile networks. This generic monitoring system is composed of monitoring agents and of data repositories called sinks that store monitored information. A centralized deployment with a single monitoring sink is supported for small networks, but DAMON provides also in a distributed scheme seamless support for multiple sinks for larger networks. In DAMON, each agent is hosted by an ad-hoc device, monitors the network behavior and sends the monitoring measurements to a data repository. DAMON defines a robust solution adapted to device mobility by supporting sink auto-discovery by agents and resiliency of agents to sink failures. The auto-discovery of repositories relies on a periodic beaconing mechanism initiated by sinks to advertise their presence and broadcast instructions to agents. A network agent receives the beacon messages,

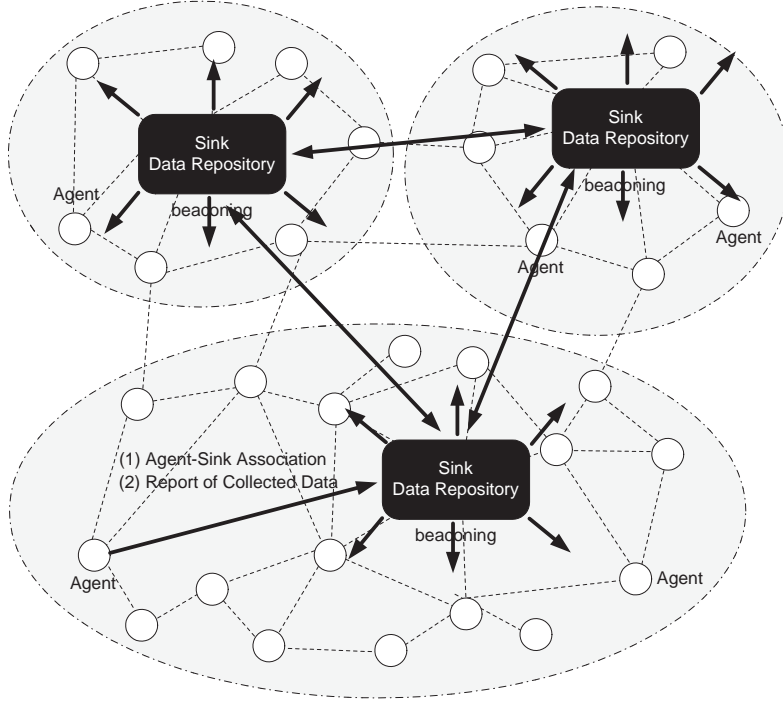


Figure 9: Distributed monitoring with DAMON

maintains a list of available data repositories and associates with the closest sink as its primary repository in order to send the collected information. The distance between the agent and the primary repository is possibly more than one hop. In this case, monitoring data from multiple agents can be aggregated by an intermediate agent to facilitate the data collection. DAMON distinguishes two types of monitoring information: time dependent information (list of neighbors, energy left on a device) which require to be frequently transmitted to sinks and time independent information (packet logs, daily traffic statistics).

The resiliency to sink failures relies on a simple switching operation to a different sink when the primary sink undergoes a failure. An agent can detect the failure of the primary sink because of the absence of beaconing messages on a given time period. In this case, the agent updates the list of available data repositories and switches to a new primary sink in order to send the collected information. In order to prevent intermittent switching between two sinks, the agent can modify its primary repository only after the successful reception of a determined number of beaconing messages. The architectural framework of a monitoring agent is composed of a packet classifier module, categorizing in input the packets according to their types and a set of packet handler modules performing specific processing on the received packets such as beacon listening

and time dependent information aggregation.

DAMON proposes a distributed scheme for monitoring ad-hoc networks based on multiple data repositories. A complete implementation of DAMON was demonstrated with the on-demand distance vector (AODV) routing protocol. But, the architecture supports a wide range of other protocols and devices. While the solution is robust and maintainable with little effort, the proximity-based association mechanism should be improved to ensure an homogeneous distribution of agents with sinks.

3.1.3 Synchronization of monitoring data

By definition an ad-hoc network is dynamic and temporal such that synchronization mechanisms are required to maintain the coherence and to maximize the life time of monitoring information in such a distributed environment. The problem of time synchronization in sparse ad-hoc networks cannot be solved with regular clock synchronization algorithms because of the restriction of node resources and of the unreliability of network links.

A synchronization method is described in [35] to synchronize information in distributed ad-hoc networks with a low message overhead as well as a low resource consumption. The solution is an adaptation of a time stamps based algorithm using unsynchronized clocks of local nodes. If the monitoring information is not duplicated, the information maintained by a local node disappears when the node is failing or is leaving the ad-hoc network. A dissemination process is therefore defined in [41] to both optimize the information survival within the network and minimize the cost of the information dissemination. A monitoring information should be defined with an expiration time as this information cannot be maintained indefinitely without any update. In particular, the statistical distribution of timescales and the impact on ad-hoc network performances is discussed in [17].

3.1.4 Processing of monitoring data

In order to process the monitoring information from an ad-hoc network, an analytical method based on distributed contrast filtering mechanisms and graph dependency analyses is proposed in [2]. The goal of this information processing is to highlight network traffic patterns in order to evaluate the network behavior. The contrast filtering can be performed in a centralized manner by the network manager but a distributed scheme where the filtering is processed locally by individual nodes is preferred for scalability and performance purposes. The filtering process aims at making in evidence disparity among nodes. Applied for a given network metric, it consists in a local comparison of the normalized value of the metric among network nodes located in a same neighborhood. A concrete usage relies on applying the contrast filter on the routed packet number per node in order to reveal routing paths and to detect network nodes interacting in the routing task.

A second method is defined in the same paper based on dependency graphs

to estimate node influence in an ad-hoc network. This more refined approach is a direct adaptation to ad-hoc networks of the work described in [8] related to the estimation of nodes influence in fixed wired networks. The method is initially inspired from research in social sciences, where social relationships are studied to detect individuals capable to influence important parts of a social network. In an ad-hoc network, the graph analysis identifies the core ad-hoc nodes participating in the network functioning. The analytical approach in [2] comes within the scope of a more global analytical framework introduced in [9] for modeling ad-hoc networks and for quantifying their behavior. However, the experimental work must be thorough with multiple scenarios implying an extensive set of network parameters.

Corrective management operations can be decided from the monitoring information. For instance in [1], a network-wide correction can be defined based on a local monitoring to improve a quality-of-service differentiated service. In this approach, each network node monitors independently the rate of high priority flow. When the rate falls out of a given threshold value, a network node is capable to signal corrective operations to the other nodes such as a selective rejection of lower priority messages in order to achieve quality-of-service guarantees. Many other configuration operations can be envisioned to optimize network performance in a close-loop management framework.

3.2 Configuration

The configuration of ad-hoc networks [36] must be performed in a self-organized manner to ensure robustness and scalability. A basic expectation in configuring ad-hoc networks lies on the autoconfiguration [40, 25] of network devices including the allocation of their IP addresses. While in a fixed wired network the device configuration is often done in a centralized manner using a dynamic host configuration (DHCP) server, such a centralized approach with fixed allocated addressed server cannot be deployed in ad-hoc networks. Alternative address allocation approaches are needed based on a distributed scheme to ensure that two nodes do not have the same address in an ad-hoc network. Allocation approaches are distinguished in [42] into three categories: conflict-free allocation, conflict-detection allocation and best effort allocation.

3.2.1 Conflict-free IP address configuration

The conflict-free IP address configuration consists in ensuring an IP address of a new node is not already assigned. This is done by using dedicated disjoint pools of addresses. A typical example of conflict-free allocation can be outlined with the IP version 6 stateless autoconfiguration. The latter relies on the statement that dedicated pools of MAC level addresses are assigned to network card producers such as an ethernet card is uniquely identified. This address can be used as a unique identifier to define the IP address in order to avoid an allocation conflict. Indeed, the stateless mechanism allows the mobile node to generate its own unique IP address using a combination of the network prefix and a suf-

fix which is locally computed based on the unique MAC address. However, this hardware-based addressing scheme does not consider that the physical addresses of a mobile node can be easily modified by reprogramming the network cards.

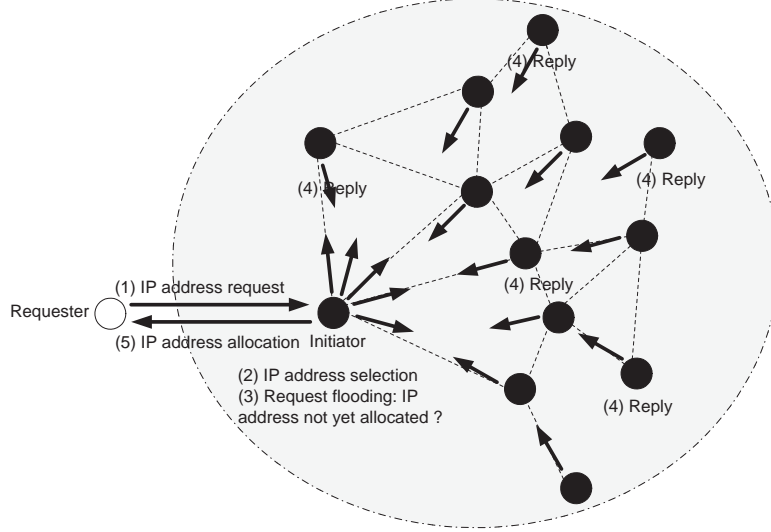


Figure 10: IP address configuration with MANETconf

3.2.2 Conflict-detection IP address configuration

The conflict-detection IP address configuration consists in selecting a supposed-to-be available IP address and assigning it after having requested a confirmation from the other network nodes that this address is effectively not in use. MANETconf [28] defines a conflict-detection distributed allocation protocol for ad-hoc networks. No centralized server can ensure the uniqueness of the allocated address. The address allocation relies therefore on a distributed mutual exclusion algorithm performed by the network nodes. Each of the latter maintains the set of the IP addresses already allocated.

The assignment process is performed in an indirect way through an intermediate node called initiator node and is defined by the following process:

- In order to obtain an IP address, a new node chooses in its neighborhood a node already part of the network and requests him to interact as an initiator.

- The initiator node selects an IP address considered to be available for use according to its local view of allocated IP addresses.
- It then performs an address query on the behalf of the new node to the other network nodes.
- At the reception of the address query, a network node sends an affirmative or negative reply if he considers the address is or is not available.
- If the initiator receives a positive reply from the network nodes, the new node is configured with the chosen address, otherwise a new address must be selected and queried by the initiator node.

MANETconf tolerates network partitioning and network merging. On one hand, the ad-hoc network partitioning is managed by an update mechanism: each time an address query is performed by a network node, the latter is capable to update the set of network nodes in his partition by identifying the nodes which did not send back any reply messages. The set of allocated addresses thus corresponds to the set of nodes part of the current partition. On the other hand, when a network merging occurs, the ad-hoc nodes from different partitions exchange and merge their set of allocated addresses. Conflicting addresses which are allocated to two nodes at the same time are identified and reconfigured by the allocation protocol.

MANETconf offers a robust protocol defining a decentralized dynamic host configuration protocol. It supports network partitioning and merging and also resolves concurrent initiation of address allocations. The solution considers a standalone ad-hoc network but could take advantage of considering an hybrid architecture where a gateway connected to an external network would be available. While MANETconf manages both message losses and node crashes, security issues such as malicious behavior of nodes remain open.

3.2.3 Best-effort IP address configuration

The best-effort IP address configuration is based on assigning by default a supposed-unused IP address to a node until a conflict is detected. The PROPHET [42] allocation defines such a best-effort scheme using the random properties of a stateful function noted $f(n)$. The stateful function is configured with an initial state value called seed and is capable to randomly generate sequences of different addresses such that the probability of having duplicate addresses is kept low. During the allocation of addresses, the function is propagated towards network nodes with different seed values as follows:

- In order to join the ad-hoc network, a new node contacts a network node and requests a new IP address.
- The node generates the new IP address using the stateful function $f(n)$ configured with its own seed. It generates also a seed value for the new node.

- The latter can in turn use the stateful function configured with this seed value to generate new IP addresses.

This distribution protocol combined with the properties of the stateful function guarantees a low probability of duplicate addresses.

PROPHET offers an allocation scheme adapted to large scale networks by defining a low communication overhead for address distribution. However, the design of the stateful function should be refined. The solution does not fully avoid address duplication and must be complemented by a conflict detection mechanism.

3.3 Cooperation Control

The deployment of an ad-hoc network is based on the cooperation of mobile nodes and on the mutual utilization of their resources. Management mechanisms are therefore required to stimulate the participation of mobile nodes, such as their involvement to share the medium access and to perform packet forwarding on the behalf of the other nodes, but also to prevent the network overload by misbehaving nodes in order to guarantee a fairness cooperation.

3.3.1 Medium access control

A misbehaving node may not respect the contention resolution mechanisms defined by a wireless medium access control protocol in order to monopolize the bandwidth resources by forcing a not equitable share of the channel. A corrective scheme is defined in [21] over the 802.11 wireless protocol to detect and handle misbehaving nodes at the medium access layer. In this wireless protocol, the contention resolution mechanism requires that a mobile node waits a backoff time interval before starting the transmission of network packets. The backoff interval value is defined by a specific random process ensuring a fair share of the channel. The misbehaving node may force a short backoff interval value such that the channel is systematically assigned to the node before the other ones. The proposed scheme defined by some modifications of the wireless protocol consists in improving the resolution mechanism by implementing some control capabilities to the node which receives the network packets.

- In a regular scenario, the sender node which wants to transmit packets to a receiver node is the node waiting during the shortest backoff interval.
- In the corrective scheme, the receiver node is capable to monitor the packet traffic generated by the sender nodes and to detect among them a misbehaving node. The receiver identifies the deviations from the protocol by monitoring the frequency and duration of the access channel by sender nodes.

If a sender node deviates from the wireless protocol, the receiver node can in a first time penalize the sender by assigning him a larger backoff interval. If the

sender node definitively does not respect the protocol, the receiver can refuse all the transmissions initialized by the misbehaving node.

The corrective scheme allows to improve a fair share of the wireless channel and facilitates the detection of misbehaving nodes by modifying the contention resolution mechanisms of the 802.11 wireless protocol. However, the solution focuses on a specific misbehavior detection and does not address for instance the case of misbehaving nodes using multiple physical addresses to acquire more bandwidth resources.

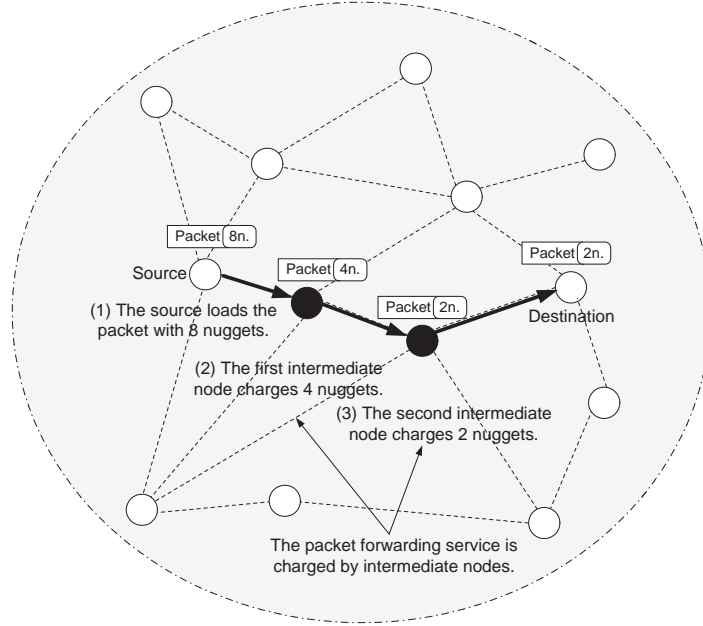


Figure 11: Cooperation stimulation with a virtual currency

3.3.2 Routing control

An efficient cooperation model cannot be limited to the medium access layer but must also be defined at the higher network layers. At the routing layer, a control scheme is introduced in [23] in order to detect and limit the impact on the routing process of mobile nodes that agree to forward network packets but fail to perform the task.

This control architecture is composed of two main components both deployed over network nodes: a watchdog capable to monitor failing network nodes and a pathrater capable to mitigate routing misbehavior.

- The watchdog detects misbehaving nodes according to the following mech-

anism. After a mobile node forwards the packets to an intermediate node, it listens to the intermediate node to check out that this node continues the forwarding process. The watchdog component maintains a buffer of the recently sent packets and compares it to the network packets emitted by the intermediate node. If a network packet is kept in the buffer more than a predefined timeout, the watchdog considers that the intermediate node fails to perform packed forwarding and sends in this case a notification message to the source node.

- The pathrater component receives the notification messages and maintains a rating of each network node to define their capability to forward network packets. High rating values are equivalent to misbehaving nodes. Both misbehavior rating information and regular link layer reliability information are combined to define a new route in the ad-hoc network.

The routing mechanism is improved by taking into account the behavior of intermediate nodes and by avoiding the definition of new routes through misbehaving nodes. However, the watchdog-pathrater model has some limitations. In particular, the misbehavior detection may be biased because of the physical environment conditions such as ambiguous collisions and limited transmission power.

3.3.3 Economical model

An economical model is proposed in [10, 11] at the service layer in order to both stimulate the cooperation and prevent the network overloading. A virtual currency called nuggets is introduced to enforce service availability and in particular to reward the nodes participating in the packet forwarding service. Two economical models are defined in this approach: the packet purse model and the packet trade model.

- In the packet purse model, the packet forwarding service is covered by the source node according to the destination node to reach. The source node loads the network packet with a sufficient number of nuggets. During the network packet route, each intermediate node charges one to several nuggets from the packet for the forwarding service. The deployment of this model is limited by the problem of avoiding an intermediate node to charge all the nuggets loaded to a packet, and also to the difficulty of estimating the number of nuggets required to reach a given destination.
- In the packet trade model, the packet is traded by intermediate nodes in an hop-to-hop manner and does not require to carry nuggets. The packet to be forwarded is processed by an intermediate node in two steps. The packet is first bought from the previous node for a given nugget price and then is sold to the next node for an higher price. In this scheme, the total price of the forwarding process is payed by the destination node and the price of the forwarding process is not required to be estimate by the source

node. However, the Packet Trade Model does not prevent a network node to overload the network by flooding.

4 Conclusions

Ad-hoc networks are a potential solution for the increasing need of mobility and ubiquitous computing. These self-configuring networks are spontaneously deployed by a set of heterogeneous nodes of various capacities and resources, without requiring any preexisting infrastructure. The dynamic and distributed nature of ad-hoc networks raises new challenges towards monitoring and controlling them. New management paradigms capable to cope with the constraints of such unreliable networks were presented in this chapter. They were completed by the description of different management application domains including network monitoring, auto-configuration of IP addresses and control of node cooperation.

From an organizational point of view, several models can be considered for the purposes of ad-hoc network management. The management approaches were classified according to the organizational model in figure 12. The approaches aim at reorganizing the management plane with different degree of decentralization. The figure shows the predominance of two antagonist models for managing ad-hoc networks:

- the centralized hierarchical model: the centralized model simplifies the management operations by considering a single point of control but suffers from communication bottleneck with the central manager. The use of intermediate nodes in a hierarchical scheme reduce the management load for the central manager.
- the distributed (flat) model : it provides better scalability performance by using a set of distributed managers. However, it requires a minimum degree of autonomy from the network nodes, such that trust and convergence issues are posed.

From a chronological point of view, the figure 12 shows the publication date of each management approach. These approaches are relatively recent, since the first cited approach is ANMP which was published in 1999. Historically, the concept of ad-hoc networking is not new, since the first infrastructure-less multi-hop network was proposed in the 1970s with the ALOHA project [24]. However, the development of telecommunications has augmented the interest in ad-hoc networking for civil applications such that an impressive number of routing protocols were proposed in the last ten years. The recent standardization of ad-hoc routing protocols clarifies the context of ad-hoc networking for defining management approaches. It defines an easier way for the piggybacking and the instrumentation of ad-hoc routing protocols for management purposes.

From a functional point of view, the areas covered by each management approach are synthesized in figure 13. The traditional classification defines

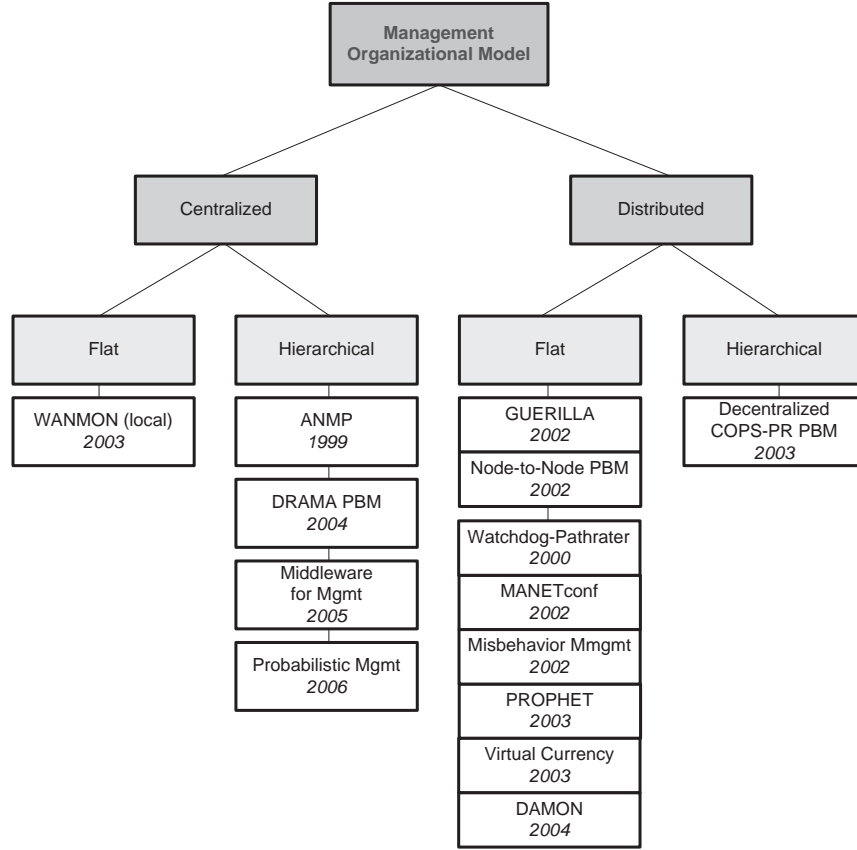


Figure 12: Classification by management organizational model

five FCAPS functional areas: fault, configuration, accounting, performance and security. Fault management is not a well-covered area. Issues related to observability are challenging: fault management can be hindered by the impossibility to observe a given node in ad-hoc networks. A node that does not reply to legitimate polling in an fixed network can be considered as not functional. In an ad-hoc network, a node might not be reachable because it is moving and is out of reachability, or because it is not functioning properly. Configuration management is a relatively well-covered area. In particular, an analytical model has been defined by [9] to quantify the scalability performance of various configuration scheme in ad-hoc networks. Accounting management is the less covered area: ad-hoc networking does not fit within traditional accounting schemes. An ad-hoc network relies on the common sharing of resources among devices such that incentive model must be preferred to stimulate the cooperation. The performance management and the security management could be further re-

searched when quality-of-service and security mechanisms will be defined in the lower network layers.

	Fault	Configuration	Accounting	Performance	Security
ANMP	✓	✓		✓	✓
GUERRILLA		✓		✓	
Decentralized COPS-PR PBM		✓		✓	
DRAMA PBM		✓		✓	✓
Node-to-Node PBM		✓		✓	
Middleware for Mgmt		✓		✓	
Probabilistic Mgmt		✓		✓	
WANMON				✓	
DAMON				✓	
MANETconf		✓			
PROPHET		✓			
Misbehavior Mgmt				✓	✓
Watchdog-Pathrater	✓			✓	
Virtual Currency			✓	✓	

Figure 13: Classification by management functional area

Ad-hoc networks are commonly considered as a particular type of networks. However, the constraints of future networks may reverse the tendency: all the optimization approaches defined for ad-hoc networking could be applied to the networks of the future. The growing dynamics of these ones might generate an edge network management crisis. Consequently as it was pointed out by [19], it is of major importance for the future to define a scalable ad-hoc routing protocol that can manage itself automatically in order to tame this crisis.

References

- [1] H. Arora and L. Greenwald. Toward the Use of Local Monitoring and Network-Wide Correction to Achieve QoS Guarantees in Mobile Ad-Hoc Networks. In *Proc. of the IEEE International Conference on Sensor and Ad Hoc Communications and Networks (SECON'04)*, Santa Clara, CA, USA, October 2004.
- [2] R. Badonnel, R. State, and O. Festor. Management of Mobile Ad-Hoc Networks : Evaluating the Network Behavior. In *Proc. of the 9th IFIP/IEEE International Symposium on Integrated Network Management (IM'05)*, pages 17–30, Nice, France, April 2005. IEEE Communications Society.
- [3] R. Badonnel, R. State, and O. Festor. Probabilistic Management of Ad-Hoc Networks. In *Proc. of the 10th IEEE/IFIP Network Operations and Management Symposium (NOMS'06)*, Vancouver, Canada, April 2006. To appear.
- [4] P. Bonacich. Factoring and Weighing Approaches to Status Scores and Clique Identification. *Journal of Mathematical Sociology*, 2:113–120, 1972.
- [5] P. Bonacich and P. Lloyd. Eigenvector-like Measures of Centrality for Asymmetric Relations. *Social Networks*, 23:191–201, 2001.
- [6] James Boney. *Cisco IOS in a Nutshell*. O'Reilly, December 2001.
- [7] S.P. Borgatti and M.G. Everett. A Graph-theoric Perspective on Centrality. *Social Networks*, 2006.
- [8] M. Burgess. *Analytical Network and System Administration. Managing Human-Computer Networks*. Number ISBN 0-470-86100-2. John Wiley & Sons, Chichester, 2004.
- [9] M. Burgess and G. Canright. Scalability of Peer Configuration Management in Logically Ad-hoc Networks. *E-Transactions on Network and Service Management (eTNSM)*, 1(1), April 2004.
- [10] L. Buttyan and J. P. Hubaux. Enforcing Service Availability in Mobile Ad-Hoc WANS. In *Proc. of IEEE/ACM Workshop on Mobile Ad Hoc Networking and Computing (MOBIHOC'00)*, Boston, MA, USA, August 2000.
- [11] L. Buttyan and J. P. Hubaux. Stimulating Cooperation in Self-Organizing Mobile Ad Hoc Networks. *ACM/Kluwer Mobile Networks and Applications*, 8(5), October 2003.
- [12] R. Chadha and H. Cheng. Policy-Based Mobile Ad Hoc Network Management for DRAMA. In *Proc. of IEEE Military Communications Conference (MILCOM'04)*, Monterey, CA, USA, October 2004.

- [13] R. Chadha, H. Cheng, Yuu-Heng Chend, and J. Chiang. Policy-based Mobile Ad-hoc Network Management. In *Proc. of the 5th IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY'04)*, New York, USA, June 2004.
- [14] W. Chen, N. Jain, and S. Singh. ANMP: Ad-Hoc Network Management Protocol. *IEEE Journal on Selected Areas in Communications (JSAC)*, 17(8):1506–1531, August 1999.
- [15] T. Clausen and P. Jacquet. Optimized Link State Routing (OLSR) Protocol. <http://www.ietf.org/rfc/rfc3626.txt>, October 2003. IETF Request for Comments 3626.
- [16] A. Deshpande, C. Guestrin, S. Madden, J. M. Hellerstein, and W. Hong. Model-Driven Data Acquisition in Sensor Networks. In *Proc. of the 13th International Conference on Very Large Data Bases (VLDB'04)*, pages 588–599, August 2004.
- [17] R. D'Souza, S. Ramanathan, and D. Temple Land. Measuring Performance of Ad-hoc Networks using Timescales for Information Flow. In *Proc. of the 22nd IEEE International Conference on Computer Communications (INFOCOM'03)*, San Francisco, CA, USA, April 2003.
- [18] R. Enns. NETCONF Configuration Protocol. Internet Draft, Internet Engineering Task Force, draft-ietf-netconf-prot-10.txt, December 2005.
- [19] B. Ford. Unmanaged Internet Protocol: taming the Edge Network Management Crisis. In *Proc. of the 2nd ACM SIGCOMM Workshop on Hot Topics in Networks (HotNets-II)*, Cambridge, MA, USA, November 2003.
- [20] S. Gouveris, S. Sivavakeesar, G. Pavlou, and A. Malatras. Programmable Middleware for the Dynamic Deployment of Services and Protocols in Ad-Hoc networks. In *Proc. of the 9th IFIP/IEEE International Symposium on Integrated Network Management (IM'05)*, pages 3–16, Nice, France, April 2005. IEEE Communications Society.
- [21] P. Kyasanur and N. Vaidya. Detection and Handling of MAC Layer Misbehavior in Wireless Networks. Technical report, Coordinated Science Laboratory, University of Illinois at Urbana-Champaign, IL, USA, August 2002.
- [22] IETF MANET (Mobile Ad-Hoc Networks) Working Group. <http://www.ietf.org/html.charters/manet-charter.html>.
- [23] Sergio Marti, T. J. Giuli, Kevin Lai, and Mary Baker. Mitigating Routing Misbehavior in Mobile Ad-Hoc Networks. In *Proc. of the IEEE/ACM International Conference on Mobile Computing and Networking (MOBI-COM'00)*, pages 255–265, August 2000.
- [24] J. M. McQuillan and D. C. Walden. The ARPA Network Design Decisions. *Computer Networks*, 1:243–289, 1977.

- [25] M. Mohsin and R. Prakash. IP Address Assignment in a Mobile Ad Hoc Network. In *Proc. of IEEE Military Communications Conference (MILCOM'02)*, volume 2, pages 856–861, October 2002.
- [26] A. Munaretto, S. Mauro, P. Fonseca, and N. Agoulmine. Policy-based management of ad-hoc enterprise networks. HP Openview University Association 9th Annual Workshop, June 2002.
- [27] C. Siva Ram Murthy and B.S. Manoj. *Ad-hoc Wireless Networks: Architectures and Protocols*. Number ISBN 0-13-147023-X. Prentice Hall (Eds.), New Jersey, USA, 2004.
- [28] S. Nesargi and R. Prakash. MANETconf: Configuration of Hosts in a Mobile Ad-hoc Network. In *Proc. of the 21st IEEE International Conference on Computer Communications (INFOCOM'02)*, New York, NY, USA, June 2002.
- [29] D. Ngo and J. Wu. WANMON: a Resource Usage Monitoring Tool for Ad-hoc Wireless Networks. In *Proc. of the 28th Annual IEEE Conference on Local Computer Networks (LCN'03)*, pages 738–745, Bonn, Germany, October 2003. IEEE Computer Society.
- [30] C. Perkins, E. Belding-Royer, and S. Das. Ad-hoc On-Demand Distance Vector (AODV) Routing. <http://www.ietf.org/rfc/rfc3561.txt>, July 2003. IETF Request for Comments 3561.
- [31] C. E. Perkins. *Ad-hoc Networking*. Number ISBN 0-201-30976-9. Pearson Education, Addison-Wesley (Eds.), New Jersey, USA, 2000.
- [32] K. Phanse. *Policy-Based Quality of Service Management in Wireless Ad-hoc Networks*. PhD thesis, Faculty of the Virginia Polytechnic Institute and State University, August 2003.
- [33] K. Phanse and L. DaSilva. Addressing the Requirements of QoS Management for Wireless Ad-hoc Networks. *International Journal on Computer Communications*, 26(12):1263–1273, July 2003.
- [34] K. Ramachandran, E. Belding-Royer, and K. Almeroth. DAMON: A Distributed Architecture for Monitoring Multi-hop Mobile Networks. In *Proc. of IEEE International Conference on Sensor and Ad Hoc Communications and Networks (SECON'04)*, Santa Clara, CA, USA, October 2004.
- [35] K. Romer. Time Synchronization in Ad-Hoc Networks. In *Proc. of ACM International Symposium on Mobile Ad-Hoc Networking and Computing (MOBIHOC'00)*, Boston, MA, USA, August 2000.
- [36] C. E. Perkins S. Singh and T. Clausen. Ad hoc network autoconfiguration: definition and problem statement. IETF draft, <http://www.ietf.org/internet-drafts/draft-singh-autoconf-adp-00.txt>, February 2004.

- [37] C.-C. Shen, C. Jaikaeo, C. Srisathapornphat, and Z. Huang. The GUERRILLA Management Architecture for Ad-hoc Networks. In *Proc. of IEEE Military Communications Conference (MILCOM'02)*, Anaheim, CA, USA, October 2002.
- [38] W. Stallings. *SNMP, SNMPv2, SNMPv3, and RMON 1 and 2*. Number ISBN 0-2014-8534-6. Addison-Wesley Professional, December 1998.
- [39] C.-K. Toh. *Ad-Hoc Mobile Wireless Networks*. Number ISBN 0-13-007817-4. Pearson Education, Prentice Hall (Eds.), New Jersey, USA, 2002.
- [40] K. Weniger and M. Zitterbart. IPv6 Autoconfiguration in Large Scale Mobile Ad-Hoc Networks. In *Proc. of the European Wireless Conference (European Wireless'02)*, Florence, Italy, February 2002.
- [41] C. Westphal. On Maximizing the Lifetime of Distributed Information in Ad-Hoc Networks with Individual Constraints. In *Proc. of the 6th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MOBIHOC'05)*, Urbana-Champaign, IL, USA, May 2005.
- [42] H. Zhou, L. NiMatt, and W. Mutka. Prophet Address Allocation for Large Scale MANETs. In *Proc. of the 22nd IEEE International Conference on Computer Communications (INFOCOM'03)*, San Francisco, CA, USA, April 2003.

HANDBOOK OF NETWORK AND SYSTEM ADMINISTRATION

CHAPTER: POLICY-BASED MANAGEMENT

Arossha Bandara, Nicodemos Damianou, Emil Lupu, Morris Sloman and Naranker Dulay
Department of Computing, Imperial College London,
180 Queensgate, London SW7 2AZ
{a.k.bandara, n.damianou, e.c.lupu, m.sloman n.dulay}@imperial.ac.uk

Abstract

Policies are rules governing the choices in behaviour of a system. They are often used as a means of implementing flexible and adaptive systems for management of internet services, distributed systems, and security systems. There is also a need for a common specification of security policy for large-scale, multi-organisational systems where access control is implemented in a variety of heterogeneous components. In this chapter we survey both security and management policy specification approaches. We also cover the issues relating to detecting and resolving conflicts which can arise in the policies and approaches for refining high level goals and service level agreements into implementable policies. The chapter briefly outlines some of the research issues that have to be solved for large-scale adoption of policy-based systems.

1. Overview

As technology pervades every aspect of the modern enterprise, from communications and information management to supply chain management and even manufacturing, the applications, services and devices used in these business activities are becoming increasingly complex. In such an environment, it is important that systems administrators are able to effectively manage the underlying technology such that the security, performance and availability requirements of the enterprise are met.

The complexity of management is caused by number of factors. Firstly, the systems to be managed will span a number of layers – from networks, servers and storage devices at the lowest level; to services and applications at the higher levels (Figure 1). To manage such systems the administrator is forced to deal with configuration, deployment and monitoring activities in a highly heterogeneous environment. Secondly, because of the range of technologies involved and the physical distribution required in global enterprises, managing these systems will be the responsibility of multiple human administrators. Making sure that the management activities of one administrator do not override or interfere with the activities of other administrators is non-trivial. In order to survive, any enterprise has to change as the world changes around it. Therefore, in order to cope with changes in the business needs of users, administrators must be able to easily change the configuration and behaviour of the systems they are managing.

Policy-based approaches to systems management are gaining widespread interest because they allow the separation of the rules that govern the behavioural choices of a system from the functionality provided by that system. This means that it is possible to adapt the behaviour of a system without the need to recode any of the underlying functionality; and changes can be applied without the need to stop and restart the system. Such features provide system administrators with the capability to manage systems in a very flexible manner.

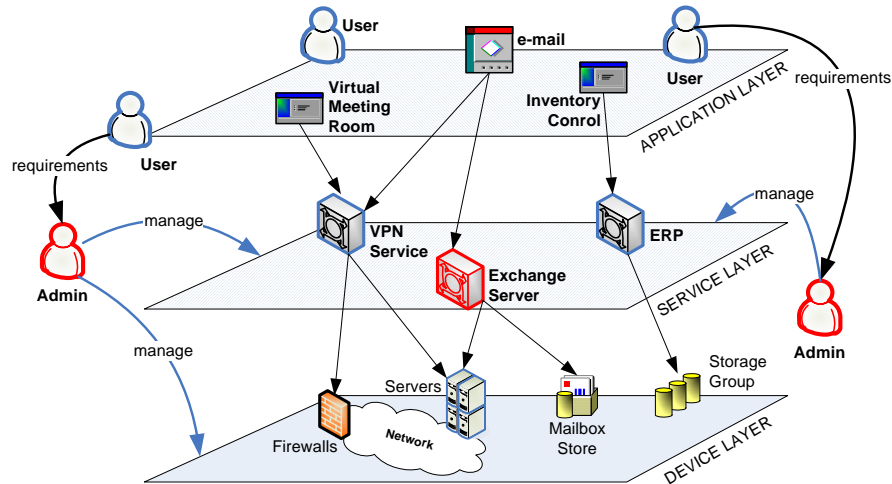


Figure 1 Managing enterprise systems

Whilst there has been research into policies and policy-based management for over a decade, there is still some ambiguity in the definition of a policy. Some researchers view policies as goals to be translated into a set of management operations; whereas others view them as event triggered rules that specify the operations to be performed if the system is in a given state (event-condition-action rules). Additionally, the term policy is used in security management to refer to rules that specify permissions and prohibitions in a managed system. Whilst, all of these definitions are valid in the context in which they were made, they are not general enough to be applied to systems management in the large. Therefore, in this chapter, we define policies as “rules governing the choices in behaviour of a system” [84]. As such, policies can take a number of different forms:

- Management policies define the conditions on which manager agents perform actions on the system to change the way it performs, or the strategy for allocation of resources. Typical policies include what resources to allocate to particular users, for example, to allocate 40% of available bandwidth for sales managers to have a video conference between 15:00 and 17:00 every Friday; when to perform back up of the critical data bases; or the conditions on which to install software. Typically management policies are implemented as obligations in the form of event-triggered condition-action (ECA) rules.
- Behavioural policies define the rules by which agents within the system interact. For example, the conditions under which they will exchange particular state information; or what filtering/transformations should be applied to email forwarded over a wireless link to a mobile user. These are also often implemented as obligations.
- Security authorisation policies are concerned with defining the conditions under which users or agents can (or cannot) access particular resources. Firewall rules are an example of authorization policy in the form of filtering rules for forwarding packets based on conditions such as source or destination IP addresses, port addresses or type of traffic. Security management policies are concerned with how the system should adapt to unusual events such as the actions to perform when a login failure is detected or what resources should be shut down when a security attack is detected. The heterogeneity of security mechanisms used to implement access control makes security management an important and difficult task.
- Routing policies are specific to routers in a network and are used to define the conditions for choosing particular paths within the network for forwarding packets. For example traffic from military installations should not be sent over insecure links, or traffic requiring low-delay should not be sent over satellite links. Some routers can also perform similar traffic filtering to that performed by firewalls.

In this survey we will concentrate on Management/Behavioural policies which are similar and security policies, particularly those related to authorization, as this is such an important issue in large-scale networked systems. Security management is covered by management policies. Routing policies are highly specialised and have been well described in a recent survey [89] so will not be described here.

Policies are persistent so a one-off command to perform an action is not a policy. Scripts and mobile agents, based on interpreted languages such as Java, can also be used to support adaptability as well as to introduce new functionality into distributed network components. Policies define choices in behaviour in terms of the conditions under which predefined operations or actions can be invoked rather than changing the functionality of the actual operations themselves. In today's Internet-based environments security concerns tend to increase when mobile code mechanisms are introduced to enable such adaptation, and so many researchers favour a more constrained form of rule-based policy adaptation.

In order to make policy based management usable in real-world applications it is important to be able to analyse policies to ensure consistency and to ensure that defined properties are preserved in the configuration of the managed system, e.g. in network quality of service management, to check that traffic marked in the same way is not allocated to different queues. Although the functionality that policies implement can be realised through general scripting languages, policy languages adopt a more succinct, declarative, form in order to facilitate analysis. Many policy languages have been proposed in the literature, but policy analysis techniques still remain poorly explored. Unless such techniques are developed and implemented in enterprise operational systems support tools, the additional expense required to deploy policy-based management will have poor short term return on investment and will remain difficult to justify.

In addition to analysing policies for consistency, administrators will need help in translating the business needs of the enterprise into policies that can be enforced by the underlying systems. Often these requirements can be expressed as high-level policies (e.g. all project teams are allowed to use the virtual meeting room application) which then must be transformed into lower-level policies, expressed in terms of management operations supported by the system. In some cases the users' requirements can be satisfied by specifying policies that perform management operations at the application layer; but other times it may be necessary to write policies that interact with the device layer of the system. This process is called policy refinement and is an area of policy based systems research that has recently gained increased attention.

Chapter Outline

This chapter provides a survey of the work on policy specification for network and security management. We begin in section 2 by looking at a general architecture for policy based management systems. This is followed by a description of the various models proposed for policy specification in section 3. In section 4 we cover some approaches to security policy specification, including logic-based languages, role-based access control and various access control and trust specification techniques. In section 5 we focus on approaches to management and behavioural policy specification, including event-triggered policies and configuration management policies. In section 6 we present an overview of frameworks that have been developed to support policy-based management techniques and in the following section we describe the Ponder policy specification language, which combines many of the above concepts and can be used for both security and management policies. Finally in section 8 we present a discussion of research issues relating to policy-based management followed by a summary of the chapter in section 9.

Online references for much of the work on policy and many of the papers described here can be found from <http://www-dse.doc.ic.ac.uk/Research/policies>.

2. General Features and Architecture

The research literature presents many proposals for implementing policy-based management systems, each with different approaches to the specification, deployment and enforcement of policies. Often these differences arise because the system is tailored to a specific type of policy, such as security authorisation policies or network routing policies. In most cases the differences manifest themselves in the form of different language constructs for specifying policy rules.

However, there are a number of components within a typical policy rule that are common to security authorization and management/behavioural obligation policies. Additionally, most approaches to policy-based management provide some capability for organising large sets of policies. In this section we present details of these common components of policy-based management systems, together with a general architecture for specifying, deploying and enforcing policies.

Policy Components

Subject defines the scope of the policy in terms of users or agents which are authorized to perform an action. For an obligation policy the subject is the agent which interprets the policy and performs the action specified. In some policy specifications the subject may not be explicitly defined in that it can be inferred from context in which the policy is specified or it relies on an administrator to assign the policy to an agent.

Target defines the scope of a policy in terms of the resources or objects on which the actions defined in the policy are permitted (for an authorization) or should be performed (for obligations). Large-scale systems may contain millions of users and resources. It is not practical to specify policies relating to individual entities, so many systems specify subjects and/or targets in terms of groups of entities. Routing and firewall policies generally perform pattern matching on source and destination addresses in order to decide how to treat a packet. Whether this corresponds to a subject or target depends on whether it is an incoming or outgoing packet and whether it is part of a request or response message.

Actions specify what must be performed for obligations and what is permitted for authorizations. Typically, actions may be one or more operations specified by an object or service interface specification. However in some systems an obligation action is a script or sequencing rules for a number of operations. Some authorization policies combine an action with a specific target object and call it a permission e.g. a read operation on the staff-list file is defined as a permission which can be assigned to specific subjects within the policy.

Conditions define the specific predicates which must be evaluated before the action can be performed. This could related to a time period e.g. between 15:00 and 17:00 on Fridays or relate to attributes of the subjects or targets themselves. For example the policy only applies when the subject is at a particular location such as in the office. All types of policies potentially can have conditions specified as part of the rule.

Triggers define events which initiate the actions in an obligation policy. An event could be generated by a component failure e.g. DoC access router failed or at a particular time e.g. at 23:59 every weekday. In some systems, the triggering event is implicit so is not explicitly defined in the policy e.g. the arrival of a packet at a router or a page request at a web server. Authorization policies are evaluated whenever an action is performed and so are not triggered by an explicit event.

Policy Organisation

Large organisations will have numerous policies and so it necessary to provide concepts for grouping and structuring policies. Many policy specification notations have some form of syntactic grouping of policies which relate to a particular application or task e.g. the policies related to registering a new user in the system, the set of policies for backing up personal workstations. There may be some common conditions which can be used for multiple policies in the group.

It is also useful to group the policies pertaining to the rights and duties of a role or position within an organisation such as a network operator, nurse in a ward or mobile computing 'visitor' in a hotel. This concept is particularly developed in role-based access control in which roles group access permissions. However the idea of a role with both authorizations (rights) and obligations (duties) is important in management applications as well and has been widely used in defining organizational structures and hierarchies.

Some systems provide support for specifying nested organizational structures which consist of instances of related roles which could represent a ward in a hospital, department in a university, branch of a bank or members of a committee. The structure may include policies pertaining to the relationships between roles or how the structure interacts with its environment.

Policy-based Management System Architecture

A typical policy-based management system consists of a policy management tool, a policy repository, a management information base (MIB), management agents, access controllers, an event monitoring system and the managed resources themselves. The relationship between these various components is shown in Figure 2.

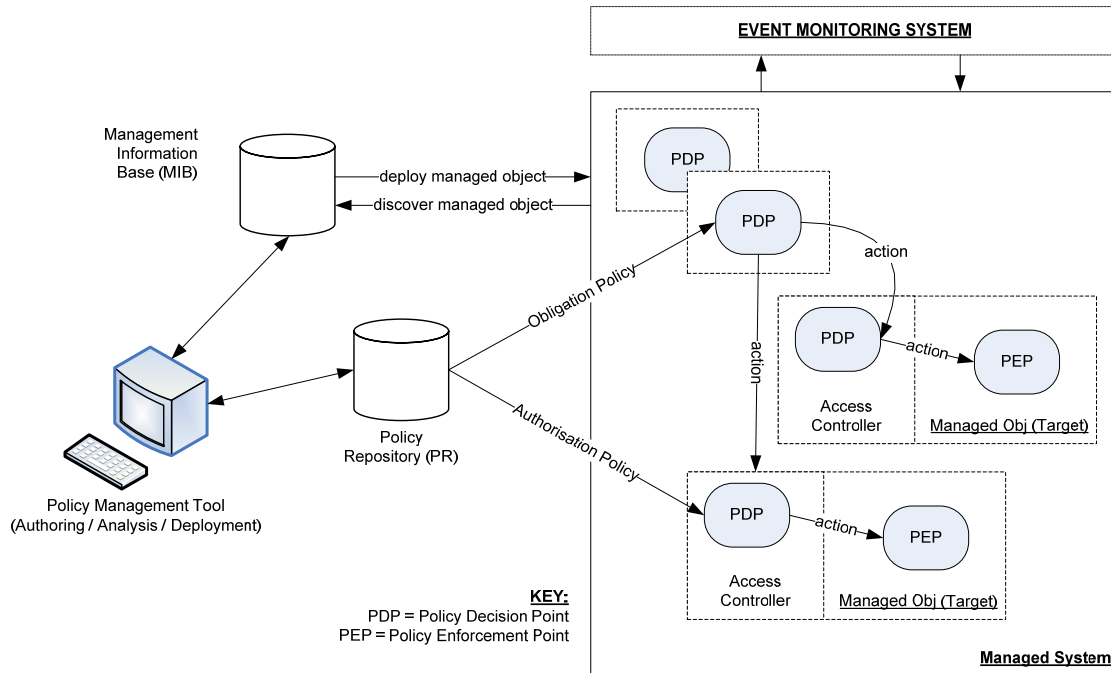


Figure 2 Typical Policy Management Implementation Framework

The policy management tool allows the administrator to create, edit, analyse and deploy policies through the policy repository. Additionally, the management tool provides an interface to the management information base (MIB), which stores the specifications of the managed resources in the system. In a typical system, the MIB will be populated using a discovery service that can detect the managed devices, services and applications in the enterprise environment. The policy management tool can also be used to deploy managed resources into the system. In recent years, a significant effort has been devoted to standardising the information models used for specifying policies and managed resources. Outcomes of this work include the Common Information Model (CIM) specification produced by the IETF/DMTF [43].

Policy implementation requires a Policy Execution Point (PEP) which actually enforces policies e.g. by deciding whether to forward a message, permit an invocation or set the priority in a packet to a particular level. This action is the result of interpreting policies which is done by a Policy Decision Point (PDP). Some legacy components may not have the capability of interpreting the policies so the PDP and PEP may be implemented as separate entities with the PEP querying the PDP for policy decisions. However in more recent systems, a combined PDP/PEP would be implemented as a management agent. A typical system would have many distributed management agents enforcing the obligation and authorisation policies. Obligation policies are interpreted by the subject management agents whereas authorisation policies are usually enforced by access controllers at the target, as discussed below. Organisational constructs, such as the roles or domains described in the previous section can be used as place holders in policy specifications, thus allowing late binding of policies to management agents.

In order to enforce authorisation security policies, the policy-based management system uses access controllers which may consist of separate or combined PDP/PEP. These are components that intercept any action invocations on a managed object and use policy rules to decide if the action should be permitted or

not. In order to make an access control decision, with each invocation, the controller needs to be provided with the identity of the requestor (subject) as well as the action itself.

The event monitoring system is responsible for disseminating events generated by managed objects to policy decision points that have been loaded with obligation policies that will be triggered by the events. In typical implementations, the event system is implemented using a publish/subscribe architecture, with a network of distributed event routers.

3. Policy Models

When deploying policy-based management in large-scale systems it is important to have models for relating the policy rules to the managed resources and users of the system. Such models can often simplify the policy specification notations used and offer opportunities for automated analysis of policy specifications for inconsistencies. In this section we describe some of the policy models proposed by the research community. We will go on to describe policy specification approaches that use some of these models in later sections.

Role-Based Access Control (RBAC) Specification

Roles permit the grouping of permissions related to a position in an organisation such as finance director, network operator, ward-nurse or physician. This allows permissions to be defined in terms of positions rather than individual people, so policies do not have to be changed when people are reassigned to different positions within the organisation. Another motivation for RBAC has been to reuse permissions across roles through a form of inheritance whereby one role (often a superior in the organisation) can inherit the rights of another role and thus avoid the need to repeat the specification of permissions.

Sandhu et al. [81] have specified four conceptual models in an effort to standardise RBAC. We discuss these models in order to provide an overview of the features supported by RBAC implementations. $RBAC_0$ contains *users*, *roles*, *permissions* and *sessions*. Permissions are attached to roles and users can be assigned to roles to assume those permissions. A user can establish a session to activate a subset of the roles to which the user is assigned. $RBAC_1$ includes $RBAC_0$ and introduces *role hierarchies* [80]. Hierarchies are a means of structuring roles to reflect an organisation's lines of authority and responsibility, and specify inheritance between roles. Role *inheritance* enables reuse of permissions by allowing a senior role to inherit permissions from a junior role. For example the finance director of a company inherits the permissions of the accounts manager, as the latter is the junior role. Although the propagation of permissions along role hierarchies further simplifies administration by considerably reducing the number of permissions in the system, it is not always desirable. Organisational hierarchies do not usually correspond to permission-inheritance hierarchies. The person in the senior role may not have the specific skills needed for the more junior role. For example, a managing director would not usually be able to perform the functions of a systems administrator much lower down in the organisational hierarchy. Such situations lead to exceptions and complicate the specification of role hierarchies [68]. Another problem with RBAC is that it does not cater for multiple instances of a role with similar policies but relating to different target objects. For example there may be two different instances of a network operator role, responsible for the North and South regions of the network respectively.

$RBAC_2$ includes $RBAC_0$ and introduces *constraints* to restrict the assignment of users or permissions to roles, or the activation of roles in sessions. Constraints are used to specify application-dependent conditions, and satisfy well-defined control principles such as the principles of least-permission and separation of duties. Finally, $RBAC_3$ combines both $RBAC_1$ and $RBAC_2$, and provides both role hierarchies and constraints. In recent work Sandhu et al. [78] propose an updated set of RBAC models in an effort to formalise RBAC. The models are called: flat RBAC, hierarchical RBAC, constrained RBAC and symmetrical RBAC, and correspond to the $RBAC_0$ – $RBAC_3$ models. Although the updated models define the basic features that must be implemented by an RBAC system more precisely, their description remains informal. A number of variations of RBAC models have been developed, and several proposals have been presented to extend the model with the notion of relationships between the roles [15], as well as with the idea of a team, to allow for team-based access control where a set of related roles belonging to a team are activated simultaneously [95].

IETF/DMTF Policy Core Information Model

The area of network policy specification has recently seen a lot of attention both from the research and the commercial communities. *Network policy* comprises rules that define the relationship between clients using network resources and the network elements that provide those resources. The main interest in network policies is to manage and control the quality of service (QoS) experienced by networked applications and users, by configuring network elements using policy rules. The most notable work in this area is the Internet Engineering Task Force (IETF) policy model, which considers policies as rules that specify actions to be performed in response to defined conditions:

```
if <condition(s)> then <action(s)>
```

The condition-part of the rule can be a simple or compound expression specified in either conjunctive or disjunctive normal form. The action-part of the rule can be a set of actions that must be executed when the conditions are true. Although this type of policy rule prescribes similar semantics to an obligation of the form event-condition-action, there is no explicit event specification to trigger the execution of the actions. Instead it is assumed that an implicit event such as a particular traffic flow, or a user request will trigger the policy rule. The IETF approach does not have an explicit specification of authorisation policy, but simple admission control policies can be specified by using an action to either allow or deny a message or request to be forwarded if the condition of the policy rule is satisfied. The following are simple examples of the types of rules administrators may want to specify. The first rule assures the bandwidth between two servers that share a database, directory and other information. The second rule gives high priority to multicast traffic for the corporate management sub-network on Monday nights from 6:00pm to 11:00pm, for important (sports) broadcasts:

```
if ((sourceIPAddress = 192.168.12.17 AND
    destinationIPAddress = 192.168.24.8) OR
    (sourceIPAddress = 192.168.24.8 AND
    destinationIPAddress = 192.168.12.17)) then
    set Rate := 400Kbps

if ((sourceIPSubnet = 224.0.0.0/240.0.0.0) AND
    (timeOfDay = 1800-2300) AND
    (dayOfWeek = Monday)) then
    set Priority := 5
```

The IETF do not define a specific language to express network policies but rather a generic object-oriented information model for representing policy information following the rule-based approach described above, and early attempts at defining a language [91] have been abandoned. The **policy core information model** (PCIM) [69] extends the common information model (CIM) [43] defined by the Distributed Management Task Force (DMTF) with classes to represent policy information. The CIM defines generic objects such as managed system elements, logical and physical elements, systems, service, users, etc, and provides abstractions and representations of the entities involved in a managed environment including their properties, operation and relationships. The information model defines how to represent managed objects and policies in a system but does not define how to actually specify policies. Apart from the PCIM, the IETF are defining an information model to represent policies that administer, manage, and control access to network QoS resources for integrated and differentiated services [85]. The philosophy of the IETF is that business policies expressed in high-level languages, combined with the network topology and the QoS methodology to be followed will be refined to the policy information model, which can then be mapped to a number of different network device configurations. Strassner shows how this approach can be applied to policy based network management in the context of work to develop the Next Generation Operations Support Systems (NGOSS) [90]. Vendors following the IETF approach are using graphical tools to specify policy in a tabular format and automate the translation to PCIM. We provide an overview of commercial tools in Section 6.

Figure 3 shows the classes defined in the PCIM and their main associations. Policy rules can be grouped into nested policy groups to define policies that are related in any application specific way, although no mechanism exists for parameterising rules or policy groups. Note that both the actions and conditions can be stored separately in a policy repository and reused in many policy rules. A special type of condition is the time-period over which the policy is valid. The *PolicyTimePeriodCondition* class covers a very complex specification of time constraints.

Policy rules can be associated with a priority value to resolve conflicts between rules. However, this approach does not easily scale to large networks with a large number of rules specified by a number of different administrators. In addition policy rules can be tagged with one or more roles.

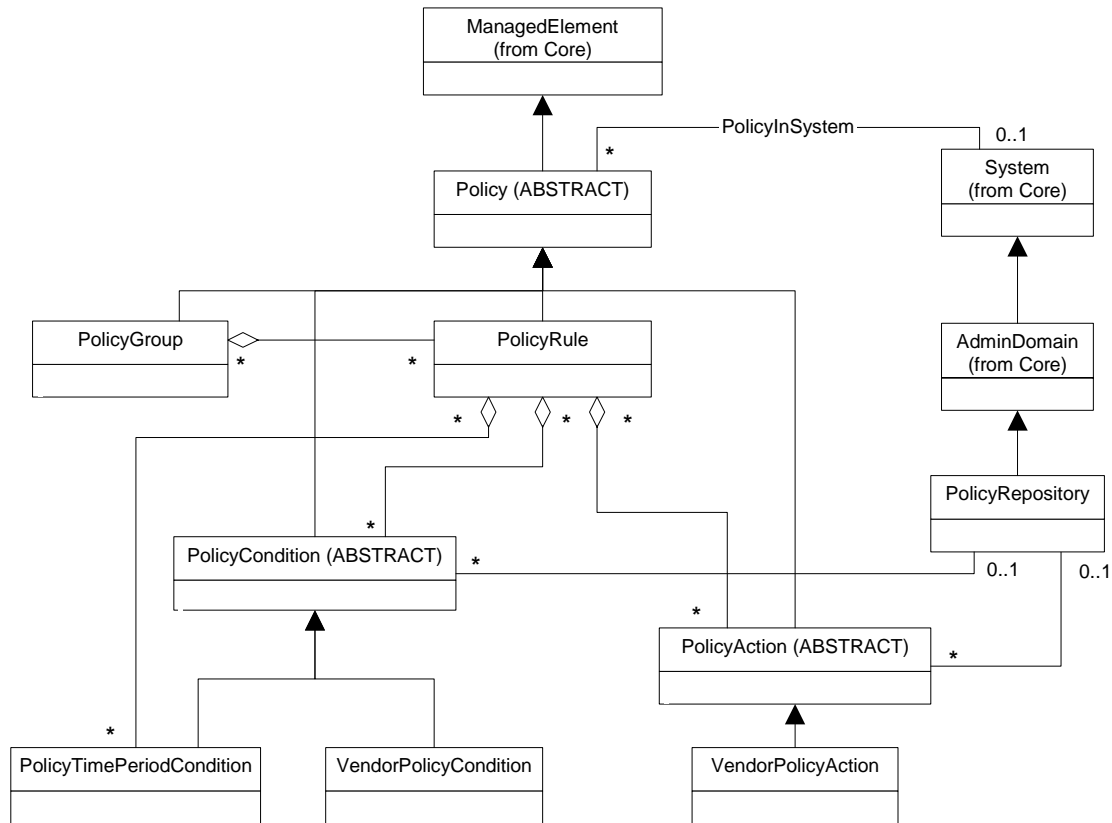


Figure 3: IETF policy core information model

An advantage of the information modelling approach followed by the IETF is that the model can be easily mapped to structured specifications such as XML, which can then be used for policy analysis as well as distribution of policies across networks. The mapping of CIM to XML is already undertaken within the DMTF [44]. The IETF define a mapping of the PCIM to a form that can be implemented in a LDAP directory [92].

```
policyID <userID> @{paths} {target} {conditions} [{action_item}]
action_item = [{condition}:] {actions}
```

Action_items in a PPL rule correspond to the *if-condition-then-action* rule of the IETF approach. The informal semantics of the rule is: “*policyID* created by *<userID>* dictates that target class of traffic may use paths only if {conditions} is true after *action_items* are performed”. The following are examples of PPL rules from [89]:


```

Policy1 <net_manager> @ {<1, 2, 5>} {class = {faculty}} {*} {priority := 1}
Policy2 <Betty> @ {<1, *, 5>} {traffic_class = {accounting}} {day != Friday :
priority := 5}

```

Policy1 states that the path starting at node 1, traversing to node 2, and ending at node 5 will provide high priority for *faculty* users. *Policy2* uses the wild-card character to specify a partial path. It states that, on all paths from node 1 to node 5, accounting class traffic will be lowered to priority 5 unless it is a Friday. In this policy the *action_items* field is used with temporal information to influence the priority of a class of traffic.

Note that the use of the *userID* is not needed in the specification of the rules, and unnecessarily complicates the grammar. The ID of the creator of a policy, as well as information such as the time of the creation, or other priority labels attached to a rule are better specified as meta-information that could be used for policy analysis. PPL does not provide any way of composing policies in groups, and there is no use of roles.

Open Distributed Programming Reference Model (ODP-RM)

The group working on the International Standards Organisation (ISO) Open Distributed Programming Reference model (ODP-RM) are defining an **enterprise language** as part of the RM-ODP Enterprise Viewpoint [52], which incorporates concepts such as policies and roles within a community. A *community* in RM-ODP terminology is defined as a *configuration* of objects formed to meet an *objective*. The objective is expressed as a contract, which specifies how the objective can be met, and a configuration is a collection of objects with defined relationships between them. The community is defined in terms of the following elements:

- The enterprise objects comprising the community,
- The roles fulfilled by each of those objects and the relationships between them,
- The policies governing the interactions between enterprise objects fulfilling roles,
- The policies governing the creation, usage and deletion of resources,
- The policies governing the configuration of enterprise objects and assignment of roles to enterprise objects,
- The policies relating to the environment contract governing the system.

Policies constrain the behaviour of enterprise objects that fulfil actor roles in communities and are designed to meet the objective of the community. Policy specifications define what behaviour is allowed or not allowed and often contain prescriptions of what to do when a rule is violated. Policies in the ODP enterprise language thus cover the concepts of obligation, permission and prohibition.

The ODP enterprise language is really a set of abstract concepts rather than a language that can be used to specify enterprise policies and roles. Recently, there have been a number of attempts to define precise languages that implement the abstract concepts of the enterprise language. These approaches concentrate on using UML to graphically depict the static structure of the enterprise viewpoint language as exemplified by [88] (see Figure 4), as well as languages to express policies based on those UML models. Steen et al. [87, 88] propose a language to support the enterprise viewpoint where policy statements are specified using the grammar shown below. Each statement applies to a role, the subject of the policy, and represents either a permission, an obligation or a prohibition for that role. The grammar of the language is concise, however it does not allow composition of policies or constraints for groups of policies. Constraints cannot be specified to restrict the activation/deactivation of roles or the assignment of users and permissions in roles. Note that the Object Constraint Language (OCL) [71] is used to express the logical conditions in the before-, if- and where- clauses defined in the grammar.

Discretionary access control (DAC) policies restrict access to objects based on the identity of the subjects and/or groups to which they belong, and are discretionary in the sense that a subject can pass its access permissions on to another subject. The notion of delegation of access rights is thus an important part of any system supporting DAC. Basic definitions of DAC policies use the access matrix model as a framework for reasoning about the permitted accesses. In the access matrix model the state of the system is defined by a triple (S,O,A) , where S is the set of subjects, O is the set of objects and A is the access matrix where rows correspond to subjects, columns correspond to objects and entry $A[s,o]$ reports the privilege of s on o . Discretionary policies do not enforce any control on the flow of information, once this information is acquired by a process, making it possible for processes to leak information to users not allowed to read it.

Mandatory access control (MAC) policies enforce access control on the basis of fixed regulations mandated by a central authority, as typified by the Bell-LaPadula, lattice-based model [17]. Lattice-based models were defined to deal with the issue of data confidentiality, and concentrate on restricting information flow in computer systems. This is achieved by assigning a security classification to each subject (an active entity that can execute actions) and each object (a passive entity storing information) in the system. Subjects and objects form a lattice based on their classification, which is used to enforce some fixed mandatory policies regarding the actions that subjects can execute on objects. The conceptual framework of the Bell-LaPadula model forms the basis of other derived models one of which is the Biba model. The Biba model uses similar controls as those used in the Bell-LaPadula model for providing *integrity* of data [19].

Non-discretionary access control (NDAC) identifies the situations in which authority is vested in some users, but there are explicit controls on delegation and propagation of authority [1]. Any global and persistent access control policy relying on access control decision information not directly controlled by the security administrator is non-discretionary. Administrative policies [79] determine who is authorised to modify the allowed access rights and exist only within discretionary policies. Whereas, in mandatory policies, the access control is determined entirely on the basis of the security classification of subjects and objects. Administrative policies can be divided into: (i) *Centralised* where a single authoriser (or group) is allowed to grant and revoke authorisations to the users. (ii) *Hierarchical* where a central authoriser is responsible for assigning administrative responsibilities to other administrators. The administrators can then grant and revoke access authorisations to the users of the system according to the organization chart. (iii) *Cooperative* where special authorisations on given resources cannot be granted by a single authoriser but needs cooperation of several authorisers. (iv) *Ownership* where a user is considered the owner of the objects he/she creates. The owner can grant and revoke access rights for other users to that object, and (v) *Decentralised* where the owner or administrator of an object can also grant other users the privilege of administering authorisations on the object.

Over the years other sophisticated security models have been proposed to formalise security policies required for commercial applications. The **Clark-Wilson model** [36] is a well known one for commercial data processing practices. Its main goal is to ensure the integrity of an organisation's accounting system and to improve its robustness against insider fraud. The Clark-Wilson model recommends the enforcement of two main principles, namely the *principle of well-formed transactions* where data manipulation can occur only in constrained ways that preserve and ensure the integrity of data, and the *principle of separation of duty*. The latter reduces the possibility of fraud or damaging errors by partitioning the tasks and associated privileges so cooperation of multiple users is required to complete sensitive tasks. Authorised users are assigned privileges that do not lead to execution of conflicting tasks. This principle has since been adopted as an important constraint in security systems.

A security policy model that specifies clear and concise access rules for clinical information systems can be found in [7]. This model is based on access control lists and the authors claim it can express Bell-LaPadula and other lattice-based models. Finally the **Chinese-wall policy** [24] was developed as a formal model of a security policy applicable to financial information systems, to prevent information flows that cause conflict of interest for individual consultants. The basis of the model is that people are only allowed to access information that is not held to conflict with any other information that they already possess. The model attempts to balance commercial discretion with mandatory controls, and is based on a hierarchical organisation of data. It thus falls in the category of lattice-based access control models.

Logic-Based Languages

Logic-based languages have proved attractive for the specification of access-control policy, as they have a well-understood formalism, which is amenable to analysis. However they can be difficult to use and are not always directly translatable into efficient implementation. This section starts by presenting some of these specification languages classified by the type of logic used in their definition. We go on to discuss languages that were specifically developed to support the Role-Based Access Control (RBAC) model.

First Order Logic

There are several examples that illustrate the application of first order logic to the specification of security policy. These include the logical notation introduced in [32]; the Role Definition Language (RDL) presented in [50] and RSL99 [4]. Because all these approaches are based on the Role Based Access Control (RBAC) model, we will defer a more detailed discussion to Section 2.2 where they can be considered together with other RBAC specification techniques.

In addition to these RBAC examples, there are some examples of the application of Z to defining security policies for a system. Z is a formal specification language that combines features of first order predicate logic with set theory [86]. In [23], the use of Z to specify and validate the security model for the NATO Air Command and Control System is described. The aim of this work was to develop a model for both discretionary and mandatory access controls based on the Bell-LaPadula model mentioned previously.

One of the main problems encountered when using first order logic for policy specification arises when negation is used together with recursive rules. This leads to logic programs that cannot be decided and cannot be evaluated in a flounder-free manner [41]. Although it is possible to avoid the use of negation or recursion, this is not practical since it diminishes significantly the expressive power of the logical language. In the next section, we discuss an alternative solution, stratified logic, which helps overcome the problems associated with using recursion and negation together.

Stratified Logic

When using first order logic based languages to describe policies, many approaches considered here are described in terms of stratified logic, which permits a constrained use of recursion and negation while disallowing those combinations which lead to undecidable programs. A stratified program is one where it is possible to order the clauses such that for any clause containing a negated literal in its body, there is a clause later in the program that defines the negated literal. Another way of describing stratified theories makes use of directed dependency graphs. These are graphs that comprise a node for each predicate symbol appearing in the program and a directed edge from the node representing any predicate that appears in the body of a clause to the node representing the predicate defined in the head of the clause. The edges are labelled positively or negatively, where a negative symbol indicates that the predicate at the tail end of the edge appear in negated form in a clause of the program. Using this technique, a program is stratified if the dependency graph contains no cycles having a negative edge.

The concepts of stratified theories and stratification were originally developed in the context of databases [31] and were later adopted into the area of logic programming as described in [8]. Programs that use stratified logic can use negation to extend their expressive power and can be evaluated in a flounder free manner. Indeed there are numerous studies that identify stratified logic as a class of first order logic that supports logic programs that are decidable [41, 53]. Moreover, such programs are decidable in polynomial time [55]. A more detailed analysis of the computational complexity and expressive power of stratified logic can be found in [41].

The **authorisation specification language** (ASL) [55] is an example of a stratified first order logic language for specifying access control policies. Authorisation rules identify the actions authorised for specific users, groups or roles, but cannot be composed into roles to provide for reusability i.e. there is no explicit mechanism for assigning authorisations to roles; instead this is specified as part of the condition of authorisation rules. Although the language provides support for role-based access control, it does not scale well to large systems because there is no way of grouping rules into structures for reusability. The following is an example of an authorisation rule in ASL, which states that all subjects belonging to group *Employees* but not to *Soft-Developers* are authorised to read *file1*.

$\text{cando}(\text{file1}, s, +\text{read}) \leftarrow \text{in}(s, \text{Employees}) \ \& \ \neg \text{in}(s, \text{Soft-Developers})$

The *cando* predicate can also be used to specify negative authorisations; the sign in front of the action in the *cando* predicate indicates the modality of the authorisation e.g. *-read* would indicate read not permitted. However, there is no explicit specification of delegation and no way of specifying authorisation rules for groups of target objects that are not related by type. A *dercando* predicate is defined in the language to specify derived authorisations based on the existence or absence of *cando* rules (i.e. other authorisations in the system). In addition, two predicates *do* and *done*, can be used to specify history-dependent authorisations based on actions previously executed by a subject. The language includes a form of meta-policies called *integrity rules* to specify application-dependent conditions that limit the range of acceptable access control policies. In a recent paper [54] the language has been extended with predicates used to evaluate hierarchical or other relationships between the elements of a system such as the membership of users in groups, inclusion relationships between objects or supervision relationship between users.

Barker [11] adopts a similar approach to express a range of access control policies using stratified clause-form logic, with emphasis on RBAC policies. According to the author, this form of logic is appropriate for the specification of access control policies mostly due to its simple high-level declarative nature. The function-free logic adopted by Barker, defines a normal clause as an expression of the following form: $H \leftarrow L_1, L_2, \dots, L_m$ ($m \geq 0$). The head of the clause, H , is an atom and L_1, L_2, \dots, L_m is a conjunction of literals that constitutes the body of the clause. If the conjunction of literals L_1, L_2, \dots, L_m is true (proved) then H is true (proved). A literal is an atomic formula or its negation and a normal theory is defined as a finite set of normal clauses. As described previously, a stratified theory extends a normal theory by eliminating some forms of “recursion-via-negation”, which makes the computation of the theory more efficient. The negation of literals is used to specify negative permissions

In [13] the authors show how policies specified in stratified logic can be automatically translated into a subset of SQL to protect a relational database from unauthorised read and update requests. The following example from [12] demonstrates their approach:

$\text{permitted}(U, P, O) \leftarrow \text{ura}(U, R1), \text{activate}(U, R1), \text{senior-to}(R1, R2), \text{rpa}(R2, P, O)$

The above clause specifies that user U has the permission P on object O if U is assigned to a role $R1$, U is active in $R1$, and $R1$ inherits the P permission on O from $R2$. This expression assumes that the following predicates have been defined: *activate*(U, R) to denote that U is active in R , *ura*(U, R) to assign user U to role R , *rpa*(R, P, O) to assign permission P on object O to role R , and *senior-to*($R1, R2$) to denote that role $R1$ is senior to $R2$.

Deontic Logic

Deontic logic was developed starting in the 1950s by Von Wright [104], Castaneda [30] and [5] by extending modal logic with operators for permission, obligation and prohibition. Known as Standard Deontic Logic (SDL), traditionally it has been used in analysing the structure of normative law and normative reasoning in law. Because SDL provides a means of analysing and identifying ambiguities in sets of legal rules, there are many examples of the application of SDL to represent legislative documents [56, 83]. An excellent overview of the applications of SDL can be found in [105].

Before looking at the details of how SDL has been applied to policy specification, it would be useful to summarise the basic axioms of the language. These are as follows:

[SDL0] Tautologies of propositional calculus

[SDL1] $O(p \rightarrow q) \rightarrow (Op \rightarrow Oq)$

If there is an obligation that p implies q , then an obligation to do p implies an obligation to do q .

[SDL2] $Op \rightarrow Pp$

If there is an obligation to do p then p is permitted.

[SDL3] $Pp \leftrightarrow \neg O\neg p$

Iff p is permitted then there is no obligation to not do p . In other words, iff p is permitted then there is no refrain policy with respect to p .

[SDL4] $Fp \leftrightarrow \neg Pp$

Iff p is forbidden then there is no permission to do p .

[SDL5] $p, (p \rightarrow q) / q$ (Modus Ponens)

If we can show that p holds and that q is implied by p , then it is possible to infer that q must hold.

[SDL6] p / Op (O-necessitation)

If we can show that p holds, then it is possible to infer that the obligation to do p also holds.

At first glance it would appear that SDL is ideally suited to specifying policy because it provides operators for all the common policy constructs like permission, prohibition (c.f. positive and negative authorisation) and obligation. However, closer examination of some of the axioms reveals inconsistencies between the definition of policy rules and the behaviour of SDL. For example, SDL2 indicates that an obligation can imply a permission and SDL3 indicates that a permission implies no obligation to not do an action. However, these implications between permissions and obligations do not exist in many systems in which the obligation and authorisation policies are specified independently and implemented in different ways.

Some of the earliest work using deontic logic for security policy representation can be found in [47]. The focus of this work was to develop a means of specifying confidentiality policies together with conditional norms.

In [33], SDL is used to represent security policies with the aim of detecting conflicts in the policy specifications. This approach is based on translating the SDL representation into first order predicate logic before performing the necessary conflict detection analysis. In an extension to this work, [38] also describes how delegation can be represented using deontic logic notation.

Ortalo describes a language to express security policies in information systems based deontic logic [72]. In his approach he accepts the axiom $Pp = \neg O\neg p$ ("permitted p is equivalent to not p being not obliged") as a suitable definition of permission. As mentioned previously, this axiom is not appropriate for the modelling of obligation and authorisation policies because the two need to be separated.

An inherent problem with the deontic logic approach is the existence of a number of paradoxes. For example, Ross' paradox can be stated as $Op \rightarrow O(p \wedge q)$, i.e. if the system is obliged to perform the action *send message*, then it is obliged to perform the action *send message* or the action *delete message*. Although there is some work that offers resolutions to these paradoxes [74], the existence of paradoxes can make it confusing to discuss policy specifications using deontic logic notations.

Role-based Security Languages

Role Definition Language

The Role Definition Language RDL [50] is based on Horn clauses and was developed as part of the Cambridge University Oasis architecture for secure interworking services. RDL is based on sets of rules that indicate the conditions under which a client may obtain a name or role, where a role is synonymous to a named group. The conditions for entry to a role are described in terms of credentials that establish a client's suitability to enter the role, together with constraints on the parameters of those credentials. The work on RDL also falls into the category of *certificate-based access control*, which is adopted by trust-management systems described separately in Section 2.3.4. The following is an example of an authorisation rule in RDL, which establishes the right for clients assigned to the *SeniorHaematologist* role to invoke the *append* method if they also possess a certificate called *LoggedOn(km, s)* issued by the *Login service*, where s is a trusted server, i.e. if they have been logged on as km on a trusted machine.

$\text{append}(\text{haematology-field}, y, x) \leftarrow \text{SeniorHaematologist}(x) \wedge \text{Login.LoggedOn}(km, s) : s \text{ in TrustedServers}$

Roles in RDL are also considered as credentials, and can be used to assign clients to other roles as in the following example where a user x , who belongs to both the *Haematologist* and the *SeniorDoctor* roles, is also a *SeniorHaematologist*:

$\text{SeniorHaematologist}(x) \leftarrow \text{Haematologist}(x) \wedge \text{SeniorDoctor}(x)$

RDL has an ill-defined notion of delegation, whereby roles can be delegated instead of individual access rights in order to enable the assignment of users to certain roles. The notion of *election* is introduced to enable a client to delegate a role that they do not themselves possess, to other clients. We believe that assignment of users to roles should be controlled with user-assignment constraints instead. The following example from [50] specifies that the *chief examiner* may elect any logged on user who belongs to the group *Staff* to be an *examiner*, for the examination subject e .

$\text{Examiner}(p, e) \leftarrow \text{Login.LoggedOn}(p, s) \triangleleft \text{ChiefExaminer} : p \text{ in Staff}$

Specifying Role Constraints

Chen et al. [32] introduce a language based on set theory for specifying **RBAC state-related constraints**, which can be translated to a first-order predicate-logic language. They define an RBAC system state as the collection of all the attribute sets describing roles, users, privileges, sessions as well as assignments of users to roles, permissions to roles and roles to sessions. They use this model to specify constraints for RBAC in two ways: (i) by treating them as invariants that should hold at all times, and (ii) by treating them as preconditions for functions such as assigning a role to a user. They define a set of global functions to model all operations performed in a RBAC system, and specify constraints which include: conflicting roles for some users, conflicting roles for sessions of some users, and prerequisite roles for some roles with respect to other users. The following example from [32] can be used as an invariant or as a precondition to a user-role assignment, to indicate that assigned roles must not be conflicting with each other:

Role set: $R = \{r_1, r_2, \dots, r_n\}$
 User set: $U = \{u_1, u_2, \dots, u_m\}$
 Check-condition: $\text{oneelement}(R) \in \text{role-set}(\text{oneelement}(U)) \rightarrow \text{allother}(R) \cap \text{role-set}(\text{oneelement}(U)) = \emptyset$

The two non-deterministic functions, *oneelement* and *allother*, are introduced in the language to replace explicit quantifiers. *Oneelement* selects one element from the given set, and *allother* returns a set by taking out one element from its input.

RSL99 [4] is another role specification language which extends the ideas introduced in [32] and can be used for specifying separation of duty properties in role-based systems. The language covers both static and dynamic separation of duty constraints, and its grammar is simple, although the expressions are rather complicated and inelegant.

Since time is not defined as part of the state of an RBAC system as defined in [32], the above languages cannot specify temporal constraints. The specification of temporal constraints on role activations is addressed in the work by Bertino et al. [18], which defines a temporal model called **TRBAC**. They propose an expression language that can be used to specify two types of temporal constraints: (i) periodic activation and deactivation of roles using periodic expressions, and (ii) specification of temporal dependencies among role activations and deactivations using role triggers.

(PE1) ([1/1/2000, ∞], Night-time, VH: activate doctor-on-night-duty)
 (PE2) ([1/1/2000, ∞], Day-time, VH: deactivate doctor-on-night-duty)
 (RT1) (activate doctor-on-night-duty \rightarrow H: activate nurse-on-night-duty)
 (RT2) (deactivate doctor-on-night-duty \rightarrow H: deactivate nurse-on-night-duty)

A role can be activated/deactivated by means of role triggers specified as rules to automatically detect activations/deactivations of roles. Role triggers can also be time-based or external requests to allow administrators to explicitly activate/deactivate a role, and are specified in the form of prioritised event expressions. The example above, from [18], shows two periodic expressions (*PE1* and *PE2*) and two role triggers (*RT1* and *RT2*).

The periodic expressions state that the role *doctor-on-night-duty* must be active during the night. The role triggers state that the role *nurse-on-night-duty* must be active whenever the role *doctor-on-night-duty* is active. In the example, the symbols *VH* and *H* stand for very high and high respectively and denote priorities for the execution of the rules.

Other Security Specification Approaches

In this section we cover other approaches to specifying security policy which includes an event-based language, the use of XML, a graphical approach and finally trust policy specification.

SPL

The **security policy language** (SPL) [75] is an event-driven policy language that supports access-control, history-based and obligation-based policies. SPL is implemented by an event monitor that, for each event, decides whether to allow, disallow or ignore the event. Events in SPL are synonymous with action calls on target objects, and can be queried to determine the subject who initiated the event, the target on which the event is called, and attribute values of the subject, target and the event itself. SPL supports two types of sets to group the objects on which policies apply: groups and categories. Groups are sets defined by explicit insertion and removal of their elements, and categories are sets defined by classification of entities according to their properties. The building blocks of policies in SPL are constraint rules which can be composed using a specific tri-value algebra with three logic operators: *and*, *or* and *not*. A simple constraint rule is comprised of two logical binary expressions, one to establish the domain of applicability and another to decide on the acceptability of the event. The following extract from [75] shows examples of simple rules and their composition. Note that conflicts between positive and negative authorisation policies are avoided by using the tri-value algebra to prioritise policies when they are combined as demonstrated by the last composite rule of the example. The keyword *ce* in the examples is used to refer to the current event.

```
// Every event on an object owned by the author of the event is allowed
OwnerRule: ce.target.owner = ce.author :: true;

// Payment order approvals cannot be done by the owner of payment order
DutySep: ce.target.type = "paymentOrder" & ce.action.name = "approve"
:: ce.author != ce.target.owner;

// Implicit deny rule.
deny: true :: false;

// Simple rule conjunction, with default deny value
OwnerRule AND DutySep OR deny;

// DutySep has a higher priority than OwnerRule
DutySep OR (DutySep AND OwnerRule);
```

SPL defines two abstract sets called *PastEvents* and *FutureEvents* to specify history-based policies and a restricted form of obligation policy. The type of obligation supported by SPL is a conditional form of obligation, which is triggered by a pre-condition event:

```
Principal_0 must do Action_0 if Principal_T has done Action_T
```

Since the above is not enforceable, they transform it into a policy with a dependency on a future event as shown below, which can be supported in a way similar to that of history-based policies:

```
Principal_T cannot do Action_T if Principal_0 will not do Action_0
```

SPL obligations are thus additional constraints on the access control system, which can be enforced by security monitors [76], and not obligations for managers or agents to execute specific actions on the occurrence of system events, independent of the access control system.

The notion of a policy is used in SPL to group set definitions and rules together to specify security policies that can be parameterised; policies are defined as classes that allow parameterised instantiation. Instantiation of a policy in SPL also means activation of the policy instance, so no control over the policy life cycle is provided. Further re-use of specifications is supported through inheritance between policies. A policy can inherit the specifications of another policy and override certain rules or sets. Policy constructs can also be used to model roles, in which case specific sets in the policy identify the users allowed to play the role. Rules or other nested policies inside a role specify the access rights associated with the role. SPL

provides the ability to hierarchically compose policies by instantiating them inside other policies, thus enabling the specification of libraries of common security policies that can be used as building blocks for more complex policies. The authors claim that this hierarchical composition also helps restrict the scope of conflicts between policies, however this is not clear, as there may be conflicts across policy hierarchies. Note that SPL does not cater for specification of delegation of access rights between subjects, and there is no explicit support for specifying roles. The following example from [75] defines an *InvoiceManagement* policy, which allows members of the *clerks* team to access objects of type *invoice*. The actual policy that permits the access is specified as *ACL* separately and instantiated within *InvoiceManagement* using the keyword *new*:

```
policy InvoiceManagement {
  // Clerks would usually be a role but for simplicity here it is a group
  team clerks ;
  // Invoices are all object of type invoice
  collection invoices =
    AllObjects@{ .doctype = "invoice" };
  // In this simple policy clerks can perform every action on invoices
  DoInvoices: new ACL(clerks, invoices, AllActions);
  ?usingACL: DoInvoices; }
```

XACML

XACML [70] is an XML specification for expressing policies for information access over the Internet, initially defined by the Organisation for the Advancement of Structured Information Standards (OASIS) technical committee (unrelated to the Oasis work at Cambridge described previously) and now being adopted as a W3C standard. The language permits access control rules to be defined for a variety of applications from protecting database entries and XML document contents to physical assets. Similar to existing policy languages, XACML is used to specify a *subject-target-action-condition* oriented policy. The notion of subject comprises identity, group, and role and the granularity of target objects is as fine as single elements within a document. The language supports both roles and groups, which are defined as collections of attributes relevant to a principal. XACML includes conditional authorisation policies, as well as policies with external post-conditions to specify actions that must be executed prior to permitting an access. e.g. “A physician may read any record and write any medical element for which he or she is the designated primary care physician, provided an email notice is sent to the patient or the parent/guardian, in case the patient is under 16”. An example of a policy specified in XACML is shown below:

```
<?xml version="1.0" encoding="UTF-8"?>
<Policy PolicyId="SamplePolicy"
  RuleCombiningAlgorithm="urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:permit-overrides">
  <!-- This Policy only applies to requests on the SampleServer -->
  <Target>
    <Subjects> <AnySubject/> </Subjects>
    <Resources>
      <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
        <AttributeValue DataType="...XMLSchema#string">SampleServer</AttributeValue>
        <ResourceAttributeDesignator DataType="http://www.w3.org/2001/XMLSchema#string"
          AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"/>
      </ResourceMatch>
    </Resources>
    <Actions> <AnyAction/> </Actions>
  </Target>
  <!-- Rule to see if we should allow the Subject to login -->
  <Rule RuleId="LoginRule" Effect="Permit">
    <!-- Only use this Rule if the action is login -->
    <Target>
      <Subjects> <AnySubject/> </Subjects>
      <Resources> <AnyResource/> </Resources>
      <Actions>
        <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
          <AttributeValue DataType="...XMLSchema#string">login</AttributeValue>
          <ActionAttributeDesignator DataType="http://www.w3.org/2001/XMLSchema#string"
            AttributeId="ServerAction"/>
        </ActionMatch>
      </Actions>
    </Target>
  </Rule>
</Policy>
```

[CONTD.]

```

<!-- Only allow logins from 9am to 5pm -->
<Condition FunctionId="urn:oasis:names:tc:xacml:1.0:function:and">
  <Apply FunctionId="... function: time-greater-than-or-equal "
    <Apply FunctionId="... time-one-and-only">
      <EnvironmentAttributeSelector DataType="... XMLSchema#time"
        AttributeId="urn:oasis:names:tc:xacml:1.0:environment:current-time"/>
    </Apply>
    <AttributeValue DataType="... XMLSchema#time">09:00:00</AttributeValue>
  </Apply>
  <Apply FunctionId="... time-less-than-or-equal "
    <Apply FunctionId="... time-one-and-only">
      <EnvironmentAttributeSelector DataType="... XMLSchema#time"
        AttributeId="... current-time"/>
    </Apply>
    <AttributeValue DataType="... XMLSchema#time">17:00:00</AttributeValue>
  </Apply>
</Condition>
</Rule>
</Policy>

```

The above policy only applies to requests for the server called "SampleServer". The Policy has a Rule with a Target that requires an action of "login" and a Condition that applies only if the Subject is trying to log in between 9am and 5pm. Although XACML supports a fine granularity of access control specification, the policy is rather verbose and not really aimed at human interpretation. XACML may be used in conjunction with SAML (security assertion and markup language) assertions and messages, and can thus also be applied to certificate-based authorisations. We discuss certificate-based authorisations in the following section. The work on XACML includes an architecture for enforcing policies which extends the IETF policy architecture described in Section 3.

SAML

The Security Assertion Mark-up Language (SAML) is the result of an initiative by Oasis to develop a notation that would support the specification and exchange of security information. One of the major design goals of SAML is to support single sign-on features that allow users to access multiple resources through a single authentication operation. A SAML document consists of a set of assertions, provided by a trusted authentication service, that specify information regarding authentication and attributes that apply to a particular subject. For example, a PEP could send a SAML document containing credential assertions obtained from a request to a PDP. The PDP would evaluate policies and send back an authorisation decision assertion, which will be used by the PEP to permit or deny access to the requested resource. An example of an SAML document fragment is presented below:

```

<saml:Assertion MajorVersion="1" MinorVersion="0" AssertionID="186CB370-5C81-
4716-8F65-
FOB4FC4B4A0B" Issuer="www.example.com" IssuedInstant="2001-05-31T13:20:00-
05:00">
  <saml:Conditions NotBefore="2001-05-31T13:20:00-05:00" NotAfter="2001-05-
31T13:25:00-05:00"/>
  <saml:AuthenticationStatement AuthenticationMethod="password"
    AuthenticationInstant="2001-05-31T13:21:00-05:00">
    <saml:Subject>
      <saml:NameIdentifier>
        <SecurityDomain>www.example.com</SecurityDomain>
        <Name>cn=Alice,co=example,ou=sales</Name>
      </saml:NameIdentifier>
    </saml:Subject>
  </saml:AuthenticationStatement>
</saml:Assertion>

```

The SAML framework supports policy specification, deployment and enforcement. However, it is designed to support access control decision-making at the implementation level of a system and does not provide any features that aid policy analysis or refinement. Beyond the example described here, SAML has general applicability. Fundamentally it encodes assertions and predefines 3 types of assertions: attribute assertions, authentication assertions, and authorisation assertions. Authorisation decisions can encode whether a principal has been authorised to access a resource, authentication assertions encode whether the principal has been authenticated (cf. single sign-on).

Trust Specification

Applications such as e-commerce and other Internet-enabled services require connectivity between entities that do not know each other. In such situations, the traditional assumptions for establishing and enforcing access control do not hold; subjects of requests can be remote, previously unknown users, making the separation between authentication and access control difficult. A possible solution to this problem is the use of digital certificates or credentials representing statements certified by trusted entities, which can be used to establish properties of their holder (e.g. identity, accreditation). Access control makes the decision of whether or not a party can execute an action based on properties that the party may have, and can prove by presenting one or more certificates. Such an approach is often called certificate-based authorisation and is adopted for the specification of trust. Trust management frameworks combine authentication with authorisation [49] and are used for applications such as web based labelling, signed email, active networks and e-commerce.

In [20, 21], two trust management applications are presented: the **PolicyMaker** and its successor **KeyNote**. Both of these applications are used to answer signed queries of the form “*does a set of requested actions r , supported by credential set C , comply with policy P ?*”, where the credentials can be public key certificates with anonymous identity. Both policies and credentials are predicates specified as simple C-like and regular expressions. In this context a policy is a trust assertion that is made by the local system and is unconditionally trusted by the system. Although trust management systems provide an interesting framework for reasoning about trust between unknown parties, assigning authorisations to keys may result in authorisations that are difficult to manage [77]. In addition, providing a common solution to both authentication and access control makes the system more complex.

The **trust policy language** (TPL) by IBM [51] provides a clearer separation between the authentication of subjects based on certificates and the assignment of permissions to those subjects which have been successfully authenticated. With TPL, the credentials result in a client being assigned to a role which has a specific set of permissions, where a role is defined as a group of entities that can represent specific organisational units (e.g. employees, managers, auditors). The assignment of access rights to roles is outside the scope of TPL; the philosophy of the work on TPL is to extend role-based access control mechanisms by mapping unknown users to well defined roles. Although the certificate is intended to be format-independent, the current implementation of the system uses X.509v3 certificates, and defines the language in XML, which makes the syntax rather verbose. Note that unlike KeyNote, TPL permits negative certificates interpreted as suggestions not to trust a user or not to assign a user to a given role. The following example is taken from [49] to demonstrate the use of TPL. In summary, the policy states that a customer of a retailer company is an employee of a department of a partner company. The first group defined is the originating *retailer*. Then, it is stated that entities having partner certificates, signed by the original retailer, are placed in the group *partners*. The group *department* is defined as any user having a partner certificate signed by the partners group. Finally, the *customer* group consists of anyone that has an employee certificate signed by a member of the departments group who has a rank greater than 3.

```
<POLICY>
  <GROUP NAME="self"> </GROUP>
  <GROUP NAME="partners">
    <RULE>
      <INCLUSION ID="partner" TYPE="partner" FROM "self"></INCLUSION>
    </RULE>
  </GROUP>
  <GROUP NAME="departments">
    <RULE>
      <INCLUSION ID="partner" TYPE="partner" FROM="partners"></INCLUSION>
    </RULE>
  </GROUP>
  <GROUP NAME="customers">
    <RULE>
      <INCLUSION ID="customer" TYPE="employee" FROM="departments"></INCLUSION>
      <FUNCTION>
        <GT>
          <FIELD ID="customer" NAME="rank"></FIELD>
          <CONST>3</CONST>
        </GT>
      </FUNCTION>
    </RULE>
  </GROUP>
</POLICY>
```

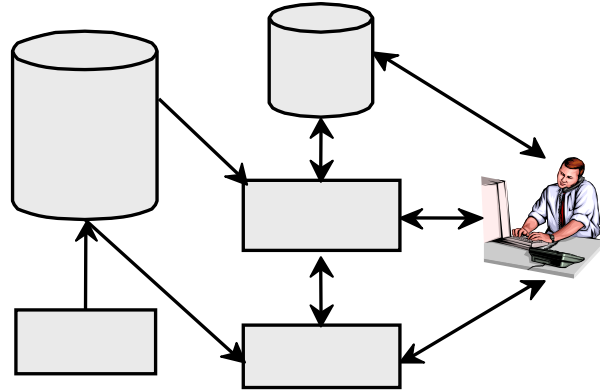


Figure 5 Sultan Trust Analysis Framework

The trust specification approaches described thus far have been based on the trust implicitly put in the issuers of credentials. More recent work by Grandison aims at calculating explicit trust based on knowledge and recommendations from others. To this end, the **SULTAN trust management framework** provides a language and analysis module for managing trust specifications [48]. The language supports conditional trust assertions of the form $\{\text{trust}(\text{Tr}, \text{Te}, \text{As}, \text{L}) \leftarrow \text{Constraints}\}$, which states that the Tr trusts/distrusts Te to perform the actions, As, at a trust/distrust level L if the optional constraints are satisfied. The specification language also allows trust relationships to be transferred through recommendation rules.

The SULTAN Trust Management Suite consists of four primary components: 1) The Specification Editor – which integrates an editor for trust specifications, a compiler for the SULTAN specifications, and auxiliary tools for storage, retrieval and translation of the specifications, 2) The Analysis Tool, which integrates a Query Command Builder, basic editor and front-end to Sicstus Prolog, 3) The Risk Service, which allows for the calculation of risk and the retrieval of risk information, and 4) The Monitoring Service, which updates information relating to experience or reputation of the entities involved in the trust relationships. The Specification Editor is used by the system administrator to encode the initial set of trust requirements and to analyse the specifications entered. This information is then stored in the system. Over time, applications present SULTAN with new information regarding the state of the system, experiences or even risk information. Whenever a decision is to be made, the SULTAN system may be consulted to provide information that will enable one to make a better decision. We will not be discussing SULTAN trust consultation in this paper. The architecture that supports this model is shown in **Error! Reference source not found.**

5. Management and Control Policies

Event-Condition-Action Rules

There has been considerable recent interest in using policies for specifying dynamically adaptable management strategies that can be easily modified to change the management approach without recoding the management system. Most of the policy-based management approaches use condition-action rules, either with or without event triggering, although some also use interpreted scripting languages. We describe these various approaches in this section.

Policy Description Language (PDL)

The **policy description language (PDL)** is an event-based language from Bell-Labs [62] that uses the *event-condition-action* rule paradigm of active databases to define a policy as a function that maps a series of events into a set of actions. The language can be described as a real-time specialised production rule system to define policies. The syntax of PDL is simple and policies are described by a collection of two

types of expressions: *policy rules* and *policy defined event propositions*. Policy rules are expressions of the form:

event **causes** action **if** condition

Which reads: If the event occurs and the condition is true, then the action is executed. Policy defined event propositions are expressions of the form:

event **triggers** policy-defined-event **if** condition

Which reads: If the event occurs under the condition, the policy-defined-event is triggered.

Events can be primitive or complex, and there are two types of primitive events: policy defined events, which are only generated by policy defined event propositions, and system events, which are generated by the environment. Primitive event classes can define attributes, and instances of the classes take actual values for those attributes that can be referenced by other events, actions or conditions within the same rule. Primitive events can be composed to form complex events that enable policies to be enforced under any of the following situations:

- If two events *e1* and *e2* occur simultaneously.
- If an event *e* does not occur.
- If an event *e2* immediately follows an event *e1*.
- If an event *e2* occurs after an event *e1*.

The following example from [59] makes use of some of the different features of the language to define a policy for a service provider network which rejects call requests when there is an excessive number of network signalling timeouts over the calls made (i.e. overload state) until the time-out rate goes down to a reasonable number. The policy has three policy defined event propositions and one policy rule proposition.

```
Events: normal_mode: policy defined event, restricted_mode : policy defined event
       call_made: system event, time_out: system event, power_on: system event

Actions: restrict_calls, accept_all_calls

Policy description:
// when the system starts the primitive event normal_mode is triggered. i.e. the
// system starts in normal mode
power_on triggers normal_mode

// when in normal_mode, a sequence of call_made or time_out events will trigger
// restrict_mode if the overload threshold is exceeded. t is the overload ratio
// of signalling timeouts over the calls made. The ^ sign denotes a sequence of
// zero or more events.
normal_mode, ^(call_made | time_out) triggers restricted_mode
if Count(time_out) > t*Count(call_made)

restricted_mode causes restrict_calls

// when in overload mode, a sequence of call_made or time_out events will
// trigger normal_mode if the normal threshold is exceeded. t' is considered to //
// be a reasonable timeout rate
restricted_mode, ^(call_made | time_out) triggers normal_mode
if Count(time_out) < t'*Count(call_made)

// Assumes only one callMade or timeOut event per epoch
normal_mode causes accept_all_calls
```

Despite its expressiveness, PDL does not support access control policies, nor does it support the composition of policy rules into roles, or other grouping structures. The language has clearly defined semantics and together with an implementation of PDL policy enforcement. Work on conflict resolution for policies written in PDL is described in [34], and extensions to the language to specify workflows for network management can be found in [59]. The language has been used to program Lucent switching products [103] and proves to be powerful in a variety of network operations and management scenarios.

Event-Trigger-Rules

Work done at the University of Florida presents a specification scheme similar to the ECA rule based approaches already discussed here, with some key differences [93]. The development of this approach,

called Event-Trigger-Rule (ETR) paradigm, is motivated by the need for rule based processing capabilities in the distributed environment of electronic commerce enterprises [93].

The ETR paradigm is a generalisation of the ECA approach where the event specification and conditions and actions of the rule are specified as separate entities. Specifying a trigger then associates the event and rule together into a policy. This is in contrast to the ECA rule specification approach, where the event specification, associated conditions and actions be combined into a single rule.

In the ETR approach, events can be classified into 3 types – method associated events, explicit events and timer events. Method events are associated with a particular method invocation and can be raised either before, after or on-commit of the method. This distinction is referred to as the coupling mode of the event. Each of these coupling modes raises synchronous events that will cause a rule to be evaluated before execution of the program continues. Additional coupling modes are instead-of (raises a synchronous event that allows the rule to replace the method invocation) and decoupled (raises an asynchronous event). Explicit events are those raised by the application during execution and timer events are those associated with a particular time of interest. The following illustrates a method associated event specification.

```

      IN   InventoryManager
      EVENT update_quantity_event(String item, int quantity)
      TYPE METHOD
      COUPLING_MODE BEFORE
      OPERATION UpdateQuantity(String item, int quantity)

```

A rule specifies some operations that should be performed if certain conditions apply. The conditional part of an ETR rule is defined as a guarded expression, where the guard is used to control evaluation of the conditional expression. This allows the entire rule to be skipped if any part of the guard expression evaluated to false, thus avoiding potential exception conditions (e.g. if required variables are not initialised). Additionally, the rule specifies an action block (cf. a ‘then’ block) and an alternative action block (cf. an ‘else’ block). The complete syntax of a rule specification is presented below.

```

      RULE rule_name(parameter list)
      RETURNS return_type]
      [ DESCRIPTION description_text]
      [ TYPE DYNAMIC/STATIC]
      [ STATE ACTIVE/SUSPENDED]
      [ RULEVAR rule_variable_declarations]
      [ CONDITION guarded_expression]
      [ ACTION operation_block]
      [ ALTACTION operation_block]
      [ EXCEPTION exception_and_handling_block]

```

When specifying a rule, it is possible to define local variables using the RULEVAR clause and also handle errors using the EXCEPTION clause. The STATE clause specifies if the rule will be active or suspended after its definition. A suspended rule will not be triggered until it is made active. The specification syntax also provides optimisation hints to the runtime environment using the TYPE clause. A dynamic rule can be changed at runtime whereas a static rule is less likely to be changed. This information is used when generating the runtime representation of the rule to provide optimal performance.

The final component of the ETR approach is the trigger. Triggers are used to specify which event(s) causes the processing of a particular rule, as illustrated in the following:

```

      TRIGGER trigger_name(parameter list)
      TRIGGEREVENT set_of_event_connected_by_OR
      [ EVENTHISTORY event_expression]
      RULESTRUC set_of_rules
      [ RETURNS return_type: rule_in_RULESTRUC]

```

The TRIGGEREVENT clause is used to specify the set of events, combined using an OR connective, that will cause the rule(s) specified in RULESTRUC to be evaluated. The event specification can be augmented using the EVENTHISTORY clause to define other event expressions that need to have occurred prior to the one defined in the TRIGGEREVENT clause. When specifying the rules to be triggered in the RULESTRUC clause, it is possible to combine several rules using one of 4 constructs: sequential (rules are triggered one after the other), parallel (rules are triggered concurrently), AND-synchronised (all members of a set of rules must

complete evaluation before another, specified, rule is triggered), and OR-synchronised (any two members of a set of rules must complete evaluation before another, specified, rule is triggered).

The literature that discusses the ETR approach presents several applications of this technique, including the development of a knowledge management network [61] and of a dynamic business process management service described in [93].

Based on its similarity to the ECA rule approaches like PDL, it is easy to see how the ETR approach could be used to specify obligation policies in a distributed system. However, because of the manner in which events are defined and the ability to associate them to method invocations, it is also possible to specify authorisation policies, albeit less succinctly, using this notation. Additionally, by separating the event specifications from the rules, the ETR approach allows the user to reuse the events in multiple triggers and thus associate them with different rules as necessary. Despite the ability to specify different types of policy, and reuse parts of the specification in multiple rules, this approach does not support other useful features like policy extension (defining policies that inherit features from some parent policy) or policy groupings (organising policies that relate to the same activity together).

Agent Interaction Policies

The Foundation for Intelligent Physical Agents (FIPA) has defined a set of use cases and abstract architectural elements that can be used to guide the specification of policy mechanisms in concrete Agent Platform architectures [46]. The objective is to define constraints on resources that agents can use, other agents with which they can interact, what messages can be exchanged, what quality of service with respect to encryption, non-repudiation etc. should be used when interacting. Their use cases cover a wide range of authorisation, obligation, delegation scenarios, identify the need for policy composition and they are also looking at including concepts of conversation policies within the FIPA work on interaction protocols.

Conversations are sequences of messages involving two or more agents intended to bring about a particular set of (perhaps jointly held) goals, and conversation policies are declarative specifications that govern specific instances of communications between agents using an agent communication language (ACL). The conversation policies could be represented as sets of fine-grained constraints on ACL usage, which define the computational process models that are implemented in agents. The main reason for conversational policies are that they provide a level of analysis that abstracts from the actual propositional content, ACL, and implementation of individual conversations. Conversational policies also help ensure reliable communication between agents whilst simplifying any inference required in determining which communicative act or other action should be made in response to a message [46].

Currently the FIPA work is at a requirements analysis and concept definition stage and they have not defined any notations for policy specification. Other approaches to policy specifications that originate from the agent systems community include KaOS, and Rei which are described in section 6. A comparative evaluation of these approaches can be found in [97].

Configuration Management

Policy-based management is also applied to *configuration management* and uses monitoring software to enable automation of network and system administration through the event-condition-action paradigm; policy-based configuration languages associate the occurrence of specified events or conditions, with responses to be carried out by an agent. **Cfengine** is a language-based administration system targeted primarily at Unix, and to a lesser extent Windows, operating systems connected via a TCP/IP network [25]. Cfengine grew out of the need to replace complex shell scripts used for the automation of administration tasks on Unix systems and allows the creation of single, central configuration file which describe how every host on the network should be configured. It uses the idea of classes to group hosts and dissect a distributed environment into overlapping sets. Host-classes are essentially labels which document the attributes of different systems. The following classes are meaningful in the context of a particular host: (i) the identity of the machine, including hostname, address, network, (ii) the operating system and architecture of the host (iii) an abstract user-defined group to which the host belongs (iv) the result of any proposition about the system, including the time or date. Policies are specified for classes of hosts and define a sequence of actions regarding the configuration of a host. The following example demonstrates the use of the language for configuration management [26]:

```

files:
  (linux|solaris).Hr12.OnTheHour.!exception_host: :
    /etc/passwd mode=0644 action=fixall inform=true

```

The first line simply defines the name *files* for the action. The second line identifies the target of the policy, i.e. all the hosts falling within the classification, the condition for execution of the policy, which is a time interval, and a trigger which specifies that the action must be executed *on the hour*.. The command-line specifies that the cfengine agent, which is always the subject of the policy, must search for all password files with an invalid mode, fix them, and inform the administrator. The class membership expression specifies all hosts which are of type *linux* or *solaris*, during the time interval from 12:00am to 12:59am, apart from a host labelled with the class *exception_host*. Policies are stored in a central repository, accessible to every host, and an active cfengine agent on each host executes the policies which apply only to that host.

Cfengine is a powerful and concise declarative scripting language, suitable for system administrators to automate common administrative tasks on Unix systems. However, it cannot be used to specify authorisation policies and lacks support for object oriented concepts such as inheritance and parameterised instantiation. Its creators admit the need for extensions to enable enterprise-level policy specification [28].

Others, focus on the specification of policies using the full power of a general purpose scripting or interpreted language (eg. TCL or Java) which can be loaded into network components or agents to implement policies. Such approaches are often leveraging the mechanisms in the area of active networks [94] to enable the control of resources at a very low level. For example Bos et al. [22] use C-programs to specify application policies for resource management in netlets, which are small virtual networks within a larger virtual network. In general for all of these approaches, the security concerns are increased, and malicious or improperly tested code can potentially damage the network. In addition, it is difficult to determine whether two computer programs specifying two different policies are contradictory or conflict with each other in any way. A comparison of different approaches to implementing policies as scripts can be found in [65].

Law-Governed Interaction (LGI)

Law-Governed Interaction [66] is another approach to defining policies for distributed heterogeneous systems. The motivation for this work is to provide a framework for controlling the flow of messages between multiple agents and thus control their interactions. In order to define the message delivery policy rules, referred to as *laws* in this approach, are specified using a simple Prolog notation. These rules are then interpreted by trusted agents, called controllers that act as proxies for every agent in the system.

In the LGI approach, the enforcement of the law is triggered by the occurrence of events that are specified as part of each law. Known as controlled events, these could occur at any point in time and the system described in the literature does not assume any a priori knowledge of how events may be generated by the systems. A law is defined as a prescription for the behaviour of the system when it is in a given state, known as the control-state, and a controlled-event occurs. This prescription for the behaviour of the system, referred to as the ruling of the law, is denoted by the sequence of primitive operations that must be performed when the law is enforced.

A law is specified with a set of rules that define the final ruling upon the occurrence of particular events in a given state. The LGI approach specifies a *do(...)* operation that is used to add primitive operations to the final ruling of the law. The following primitive operations and controlled-events are specified in the literature.

<code>sent(x, m, y):</code>	this event occurs when agent x, sends a message m, to agent y.
<code>forward(x, m, y):</code>	this operation is performed when the law being enforced by agent x rules that message m should be sent to agent y.
<code>arrived(x, m, y):</code>	this event occurs when message m, sent by agent x, arrives at agent y.
<code>deliver(x, m, y):</code>	this operation is performed when the law being enforced by agent y rules that the message m, sent by agent x, should be delivered to agent y.

Of course this set can be extended to suit the particular domain in which LGI is being applied. For example, a simplified token ring protocol law specification presented in [66]. As shown below, the set of

primitive operations is extended to include `recant(token)`, which causes the agent to clear the token property from its control state; and `set(token)`, which adds the token to the control state of the agent. Additionally, a predicate `next(D)` is used which is defined to hold when `D` is the next agent in the ring. Note that the `@` symbol is used to indicate that the predicate is to be evaluated with respect to the control state of the agent.

```
R1: sent(S, M, r) :- token@, do(forward(S, M, r)).
    {To send a message to r a node must have the term token in its cState}

R2: arrived(S, M, r) :- do(deliver(t4)).
    {Any message that arrives at r is delivered.}

R3: sent(S, yourTurn, D) :- token@, next(D@,
                             do(recant(token)),
                             do(forward(S, yourTurn, D))).
    {The owner of the token can give it up by sending the yourTurn message to the next object on the ring}

R4: arrived(S, yourTurn, D) :- do(set(token)),
                             do(deliver(memo(yourTurn))).
    {When the yourTurn message arrives at an object, the token is added to it, and an appropriate memo is delivered.}
```

In this specification, R1 states that if the `forward(...)` operation is added to the ruling and the token is present in the control state of the agent, the `sent(...)` event is generated. R2 is used to specify the policy that any message that arrives at a recipient agent is delivered. R3 specifies that in order for the `yourTurn` message to be sent (...), the destination `D` must be the next node in the ring, the current agent must have the token and the law must prescribe that the agent `recant(...)` the token and `forward(...)` the `yourTurn` message. Finally, R4 specifies that the `arrived(...)` event is generated once the token is `set(...)` into the control state of the recipient agent and the `deliver(yourTurn)` operation has been added to the ruling of the law.

In this approach, permissions and prohibition are specified as a set of rules that are similar to positive and negative authorisations. Additionally, the approach supports a common global set of constraints, similar to obligation policies, which are implemented by means of filters in every node that check that all interactions are consistent with a global law [67]. There are other examples of the use of LGI systems to specify business rules in electronic commerce applications [99] and to support the provision of security policies in heterogeneous systems [98].

6. Policy-based Management Frameworks

In addition to the logic-based policy specification approaches described in the previous sections, a number of higher-level policy languages and frameworks have been developed to support a variety of management applications. These range from systems that provide policy-based services for distributed agent environments to those that use policies for network management. A notable feature that distinguishes these approaches from the formal ones described above is that, in most cases, they combine a policy specification language with a framework for specifying the properties of the managed system and a means of deploying and enforcing policies on the managed objects. In this section we present an overview of a number of these high-level languages and frameworks, and in each case discuss how they support policy analysis and refinement.

Policy Middleware for Autonomic Computing (PMAC)

The Policy Middleware for Autonomic Computing (PMAC) framework has been developed as part of IBM's autonomic computing initiative [2]. The framework consists of a policy editing tool, a federator, an autonomic manager and a managed resource (**Error! Reference source not found.**). Policies specified using the editor are published to the federator which acts as a publish/subscribe hub for distributing policies to the relevant autonomic managers. The principal component in this framework is the autonomic manager, which is responsible for providing a local policy-based control loop to manage the resource assigned to it. Managed resources provide two interfaces, referred to as *sensors* and *effectors*, which

respectively represent the attributes that can be read from the resource and the management operations that can be performed to change the state of the resource. Since an autonomic manager can be treated as a managed component, it also provides an interface to its sensors and effectors for use by other autonomic managers.

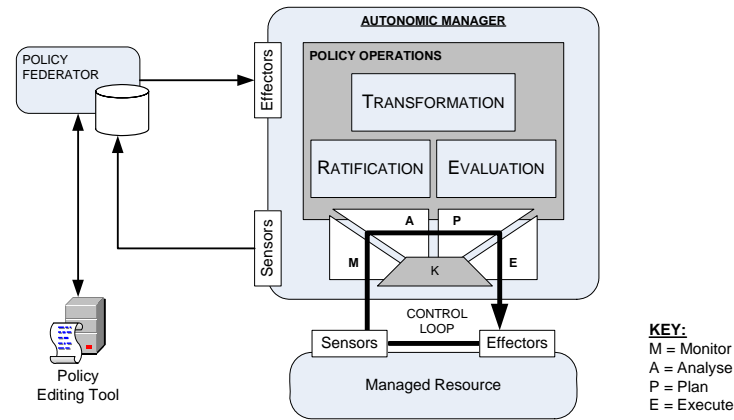


Figure 6: Policy Management for Autonomic Computing (PMAC) framework

Policies in the PMAC framework are specified as Event-Condition-Action (ECA) rules and their structure is defined using an XML schema. Additionally, the framework includes a general constraint specification language, also specified using XML, which can be used in the ECA policy rules to define the condition clause. The use of XML schema based definitions for the policy and constraint language allows the PMAC framework to extend the set of types supported by the language by simply including off-the-shelf XML definitions which come with pre-defined syntax and semantics. The problem with using an XML representation for policy is that the specifications tend to be quite verbose and not easily interpreted by the user. This problem is addressed in the PMAC framework by providing a *Simple Policy Language (SPL)* which supports the specification of ECA rules. At present, there is no documentation regarding the syntax of SPL, but it is expected to be similar to the obligation policy syntax of the Ponder framework.

PMAC provides support for policy analysis and conflict resolution using a process called *Policy Ratification* [3]. Essentially this process provides the administrator with information about the impact of a new policy on the existing set of policies. The ratification process consists for four domain-independent, generic operations: dominance checking, conflict checking, coverage checking and consistent priority assignment.

Dominance checking determines if there is a policy in the system that is subsumed (or dominated) by another. For example a policy that specifies that “password length > 5” is dominated by one that states “password length > 8” since the latter policy makes the former redundant. Generally it can be stated that a policy P1 is dominated by policy P2, if the constraint of policy P1 logically implies the constraint of P2.

One of the main challenges in policy analysis is in dealing with the constraints that control the applicability of a policy and identifying the situations in which two policies have constraints which are satisfied at the same time. The *Conflict checking* operation in PMAC is used to identify policies whose constraint statements are simultaneously true, and therefore may cause a problem if the actions defined in those policies are incompatible with each other. For example a policy that defines that “Joe has access to the database between 9am and 5pm” would be in conflict with a policy that states “Joe should not be granted access to the database on weekends” since, if both these policies are enforced, Joe is both granted and refused permission between 9am and 5pm at weekends. PMAC addresses the conflict detection requirement by recognising that the key operation in detecting a potential conflict is to determine whether the conjunction of two Boolean expressions can be satisfied. To this end, the framework implements a number of analytical algorithms for checking the satisfiability of different classes of Boolean expressions [3]. Whilst this approach is effective at detecting potential conflicts in a policy specification, since the analysis does not use a model of the system behaviour, it cannot take into account changes in system state caused by enforcing policies.

Coverage checking allows an administrator to ensure that for a given range of input parameters there is at least one policy that is applicable. This is achieved by checking whether a disjunction of Boolean expressions (e.g. the constraints of the policies) implies another (e.g. the input parameter range).

Finally, the PMAC framework assumes that problems identified through the above ratification operations will be resolved by the administrator marking policies as inactive or assigning some relative priority to them. In this latter case there is the possibility that introducing a new policy requires that existing priority values are reassigned so that the overall ordering of the policy rules remains consistent. The fourth ratification operation, *consistent priority assignments* automates this priority assignment process using a modified version of an algorithm used to insert items into an ordered list.

Policy refinement is supported in PMAC through the definition of transformation rules. These take the form “if certain conditions on a policy are satisfied (e.g. if the Role is WindowsSecurity) then update a certain portion of the policy with a new entity (e.g. replace the Condition and the Action by a new Boolean expression).” Since these transformation rules are implemented as policy rules that operate on the policies themselves, they can be executed using the same enforcement system as used by any other PMAC policy. One drawback of this approach is that the transformation rules have to be defined manually by an expert in the application domain and there is no means of verifying that the transformed policy satisfies the original goal of the user.

To summarise, the PMAC framework provides a language for specifying policy and a policy deployment and enforcement architecture. The system has been designed to be extensible through the use of XML schemas and is capable of policy analysis and refinement operations. However, the analysis process does not take into account the behaviour of the managed system and therefore cannot detect inconsistencies that are caused because the enforcement of one policy causes a state change such that some other policies then conflict. Finally, the policy refinement process is defined in terms of explicit transformation rules which must be specified by a domain expert. Therefore it is not possible to verify the correctness of the policies generated by applying the transformation rule with respect to the original policy.

KaOS

KaOS is a collection of component-based agent services developed to support a variety of mobile agent platforms [100]. Since its initial presentation KaOS has been adapted to general purpose grid computing and Web Services environments as well. The principle components of the KaOS framework are the domain service and policy service.

The KaOS policy service is responsible for specification, management, conflict resolution and enforcement of policies. In the initial versions of KaOS, policy specification takes an ontology based approach and policies are represented using the DAML¹ notation [40] but the language has since evolved to use OWL². The ontology definition makes a distinction between authorisation policies and obligation policies and each of these policy types are specialised further using a modality operator. A *positive authorisation* policy defines actions that are permitted when given constraints hold true and a *negative authorisation* policy defines actions that are prohibited under given constraints. Similarly, *positive obligations* define a requirement that a given operation be performed whereas *negative obligations* waive the requirement to perform the action (c.f. Ponder refrain policies). The basic policy ontology can be extended to support application specific concepts by defining management operations and their associated parameters.

The framework further comprises a KaOS Policy Administration Tool, that provides support for policy specification, manages ontologies, deploys policies and detects conflicts between policies in a specification. This addresses the problem associated with the verbosity of policies specified using the DAML+OIL notation, which can be hard for users to understand. This is illustrated by the example shown below, taken from [97] which specifies that members of a domain A are prohibited from performing encrypted communication with anyone outside this domain. In this example, the action is specified in lines 1-15 and the prohibition in lines 16-21.

¹ DARPA Agent Markup Language

² Ontology Web Language

KaOS provides support for detecting and resolving modality conflicts which can arise if two policies with opposite modalities specify the same actions and are enforced simultaneously. To achieve this, KaOS uses extensions to its integrated Java Theorem Prover (JTP) to implement an algorithm that is based on using the subsumption mechanisms between classes. Detected conflicts are then resolved through priority assignment and the automated creation of *harmonization policies* [100].

Because the KaOS framework has a built-in theorem prover, it is also possible to perform other types of query (e.g. “which entities have read/write access to the examination marks database”). However, as in the case of PMAC, the policy analysis procedure does not account for the behaviour of the managed system and therefore is not able to detect conflicts due to changes in the system state.

The deployment architecture makes use of KaOS domain services to determine the entities to which a policy applies. Additionally, the deployment model has a *directory service* for storing policies, while *guards* interpret policies before passing them on to *enforcers* which translate the policy actions into platform-specific operations.

```

1  <daml : Class rdf: ID="ExampleAction">
2    <rdfs: subclassOf rdf: resource="#EncryptedCommunicationAction" />
3    <rdfs: subclassOf>
4      <daml : Restriction>
5        <daml : onProperty rdf: resource="#performedBy" />
6        <daml : toClass rdf: resource="#MembersOfDomainA" />
7      </daml : Restriction>
8    </rdfs: subclassOf>
9    <rdfs: subclassOf>
10     <daml : Restriction><daml : onProperty rdf: resource="#hasDestination" />
11     <daml : toClass rdf: resource="#notMembersOfDomainA" /></daml : Restriction>
12  </rdfs: subclassOf>
13 </daml : Class>
14 <policy: NegAuthorizationPolicy rdf: ID="Example">
15   <policy: controls rdf: resource="#ExampleAction" />
16   <policy: hasSiteOfEnforcement rdf: resource="#ActorSite" />
17   <policy: hasPriority>10</policy: hasPriority>
18   <policy: hasUpdateTimeStamp>4237445645589</policy: hasUpdateTimeStamp>
19 </policy: NegAuthorizationPolicy>

```

By using the idea of platform-specific enforcement components in the runtime architecture, KaOS policies can be used in a variety of applications. Indeed, KaOS has been applied in a range of application scenarios, ranging from web services management to grid computing. It supports authorisation and management/behavioural policies and provides tools for specifying these policies, analysing them and deploying them into the managed system. Its ontology based approach allows application specific information about the managed system and operations to be added easily. However, whilst the analysis procedure does allow detection of modality conflicts and checking other properties, it does not account for policy interactions where enforcing one policy causes conflicts to arise between others. Finally, KaOS does not provide any support for policy refinement.

Rei

Rei was developed by Kagal et al., to support policy-based management in pervasive computing applications [57]. It provides three language constructs: policy objects, meta-policies and speech acts. Policy objects are defined as rights, prohibitions, obligations or dispensations which conceptually correspond to the positive and negative authorisations and positive and negative obligation policies described in KaOS and Ponder.

Policies are specified using an ontology-based approach to represent each policy type and therefore policy rules are specified in the verbose OWL representation [73], which is similar to the notation used by KaOS. However, Rei also provides a Prolog based notation that is more compact and easier to understand. The

basic ontology includes the description of actions in terms of an action identifier, the target objects on which the action can be performed, the set of pre-conditions required to perform the action and the post-condition of the action.

The listing below shows an example authorisation policy that specifies that any final year student can print on the colour printer. The policy is defined using the action `printColourPage` (Line 1) which has the pre-conditions that the printer has paper and has toner; and the post-condition that there is one less page in the paper tray. This is used in a `has(Object, PolicyDef)` predicate to specify that Students who are members of the `finalYearGroup` have the right to perform the action (Line 2)

```
1  action(printOnePageColour, [printerColour],
      (containsCartridge(printerColour),
       availablePaper(printerColour, X), X>1),
       availablePaper(printerColour, X-1))

2  has(Student, right(printOnePageColour, isMember(Student, finalYearGroup))).
```

The Rei system supports policy enforcement by providing a policy decision engine that deduces the rights and obligations of objects in the managed system in response to requests that specify the current state of the system. The use of Prolog to specify policy rules means that Rei can use a Prolog reasoning engine to analyse the policy specification and detect modality conflicts. Since the policy engine only supports deductive reasoning, only runtime conflicts can be detected. By using the ontology to define information about application specific actions, theoretically it should be possible for the analysis process to detect application specific conflicts. For example, an obligation to lock a door would conflict with another obligation that required the door be opened. However, in the literature there are no examples of such conflict analysis being performed on Rei policies.

As mentioned, the Rei language also includes the concepts of speech acts [82] and meta-policies. Speech acts are used to allow objects in the system to dynamically transfer rights and obligations to other entities. This is similar to the idea of delegation discussed in the literature. Meta-policies are rules about other policies and are used to resolve conflicts that are detected by the Rei policy engine at runtime.

Whilst Rei has been successfully used in a number of pervasive computing applications to provide policy based access control functionality, it does have a number of limitations. Firstly, since the conflict detection method depends on deductive reasoning, it only works when a complete definition of the system state is provided, i.e. at runtime. Whilst this is useful, it would be better if administrators were able to detect potential conflicts in their policies at specification time so that they can minimise the risk of deploying a policy that causes a failure in the system. Other limitations are the lack of support for policy refinement and the absence of any tools to help administrators specify policy rules.

Commercial Frameworks

The majority of policy-based management frameworks, and associated tools, are research prototypes and there are few examples of policy-based management being applied in a commercial environment. Verma [101] describes a QoS tool used to specify Service Level Agreements (SLAs) and to manipulate SLA related information in a tabular format. The tool transforms high-level policy information into device configurations, and stores them in an LDAP directory. Another tool, presented in [29], focuses solely on template-based refinement of policies from high-level goals.

Existing work within the RBAC community is limited to specifying access control configurations in terms of roles. A centralised tool, presented in [96], translates access control configuration from the RBAC framework to the target's native security mechanism, which is then transported to the target. Another web-based tool, presented in [14], allows administrators to specify roles, role hierarchies and constraints to implement RBAC for networked servers using Web protocols in order to manage access to an organisation's Web information.

In policy-based networking most of the tool support comes from industry and is based on the IETF policy framework. The majority of the commercial tools are specific to quality of service management, but many also include access control configuration. The list of vendor products is very big and space considerations

prevent us from discussing each tool in detail. However, because these tools have a significant influence on the adoption of policy-based management solutions, we feel it is necessary to consider some of the major commercial policy-based network management products in greater depth.

It should be noted that the information presented here is based on public-domain documentation, available at the time of writing, from the vendors' website and industry surveys. For the latest information, surveys of commercial tools, together with product comparisons are available on the web (see <http://www-dse.doc.ic.ac.uk/Research/policies> for more information).

HP Openview PolicyXpert

HP's PolicyXpert tool is a multi-platform policy based management solution, designed for integration into the company's Openview network management suite. In its current release, version 2.1, PolicyXpert supports traffic management actions ranging from priority marking to DiffServ code points. Like many of the other tools considered here, policies are defined using the *if <condition> then <action>* paradigm where conditions can be based on packet information, time of day or higher-level protocol information like HTTP URL or VLAN ID. The tool supports many of the prevalent standards, including COPS, DiffServ and RSVP.

Cisco CiscoAssure

Cisco's policy based management offering, CiscoAssure Policy Manager, is also aimed at QoS service management. Although policies are specified using the condition/action approach defined by the IETF-CIM standard, the tool policies themselves are stored in a flat-file database [35]. The user interface allows administrators to easily specify multiple conditions for triggering policies. Like the other tools considered here, conditions can be specified using a combination of IP addresses (source and destination), application ports, and the protocol being used (IP, TCP or UDP). Policy actions are applied to routers by using the Cisco Command Line Interface (CLI) language. Multi-vendor interoperability is provided with an implementation of COPS. In addition to supporting QoS related management operations, this tool allows the administrator to define access control policies for the devices being managed.

Allot Communications NetPolicy

NetPolicy aims to provide policy based management capabilities for a range of Allot Communication's network hardware in addition to Cisco routers. However, results of tests performed on an early version of this tool concluded that the Cisco support was incomplete [6]. Once again, policies are specified using the condition/action notation, and the conditions can be defined in terms of the packet information parameters mentioned previously. The policy repository is implemented using LDAP and policy information is passed to target devices using either COPS or CLI. Additionally, NetPolicy supports management operations on simple access control lists.

7. Ponder Policy Framework

The Ponder language for specifying Management and Security policies [39] evolved out of work on policy management at Imperial College over a period of about 10 years. Ponder is a declarative, object-oriented language that can be used to specify both security and management policies. Ponder authorisation policies can be implemented using various access control mechanisms for firewalls, operating systems, databases and Java [37]. It supports obligation policies that are event triggered condition-action rules for policy based management of networks and distributed systems. Ponder can also be used for security management activities such as registration of users or logging and auditing events for dealing with access to critical resources or security violations. Key concepts of the language include domains to group the object to which policies apply, roles to group policies relating to a position in an organisation [63], relationships to define interactions between roles and management structures to define a configuration of roles and relationships pertaining to an organisational unit such as a department.

Domains

Domains provide a means of grouping objects to which policies apply and can be used to partition the objects in a large system according to geographical boundaries, object type, responsibility and authority or

for the convenience of human managers. Membership of a domain is explicit and not defined in terms of a predicate on object attributes. A domain does not encapsulate the objects it contains but merely holds references to objects. A domain is thus very similar in concept to a file system directory but may hold references to any type of object, including a person. A domain, which is a member of another domain, is called a **sub-domain** of the parent domain. A sub-domain is not a subset of the parent domain, in that an object included in a sub-domain is not a *direct* member of the parent domain, but is an *indirect* member, c.f., a file in a sub-directory is not a direct member of a parent directory. An object or sub-domain may be a member of multiple parent domains i.e. domains can overlap.

An advantage of specifying policy scope in terms of domains is that objects can be added and removed from the domains to which policies apply without having to change the policies. Domains have been implemented as directories in an extended LDAP Service.

Ponder primitive policies

Authorisation policies define what activities a member of the subject domain can perform on the set of objects in the target domain. These are essentially access control policies, to protect resources and services from unauthorized access. A positive authorisation policy defines the actions that subjects are permitted to perform on target objects. A negative authorisation policy specifies the actions that subjects are forbidden to perform on target objects.

The language provides reuse by supporting the definition of policy types to which any policy element can be passed as a formal parameter. Multiple instances can then be created and tailored for the specific environment by passing actual parameters as shown below .

```
type auth+ PolicyOpsT (subject s, target <PolicyT> t) {
    action load(), remove(), enable(), disable() ; }

inst auth+ switchPolicyOps=PolicyOpsT(/NetworkAdmins, Nregion/switches);
inst auth+ routersPolicyOps=PolicyOpsT(/QoSAdmins, /Nregion/routers);
```

The two policy instances created from a PolicyOpsT type allow members of /NetworkAdmins and /QoSAdmins (subjects) to load, remove, enable or disable objects of type PolicyT within the /Nregion/switches and /Nregion/routers domains (targets) respectively.

Policies can also be declared directly without using a type as shown in the negative authorisation policy below, which indicates the use of a time-based constraint to limit the applicability of the policy

```
inst auth- /negativeAuth/testRouters {
    subject /testEngineers/trainee ;
    action performance_test() ;
    target <routerT> /routers ;
    when time.between ("0900", "1700")
}
```

Trainee test engineers are forbidden to perform performance tests on routers between the hours of 0900 and 1700. The policy is stored within the /negativeAuth domain.

Ponder also supports a number of other basic policies for specifying security policy: *Information filtering* policies can be used to transform input or output parameters in an interaction. For example, a location service might only permit access to detailed location information, such as a person is in a specific room, to users within the department. External users can only determine whether a person is at work or not. *Delegation* policies permit subjects to grant privileges, which they possess (due to an existing authorisation policy), to grantees to perform an action on their behalf e.g., passing read rights to a printer spooler in order to print a file. *Refrain* policies define the actions that subjects must refrain from performing (must not perform) on target objects even though they may actually be permitted to perform the action. Refrain policies act as restraints on the actions that subjects perform and are implemented by subjects. See [39] for more details and examples of these policies.

Obligation policies are event-triggered condition-action rules, similar to Lucent's PDL, and define the activities subjects (human or automated manager components) must perform on objects in the target domain.

```

inst oblig loginFailure {
  on
  subject
  target <userT>
  do
    3*loginfail(userid) ;
    s = /NRegion/SecAdmin ;
    t = /NRegion/users ^ {userid} ;
    t.disable() -> s.log(userid) ;
}

```

This policy is triggered by 3 consecutive loginfail events with the same userid. The NRegion security administrator (SecAdmin) disables the user with userid in the /NRegion/users domain and then logs the failed userid by means of a local operation performed in the SecAdmin object. The '->' operator is used to separate a sequence of actions in an obligation policy. Names are assigned to both the subject and the target. They can then be reused within the policy. In this example we use them to prefix the actions in order to indicate whether the action is on the interface of the target or local to the subject.

Events can be simple, i.e. an internal timer event, or an external event notified by monitoring service components e.g. a temperature exceeding a threshold or a component failing. Composite events can be specified using event composition operators.

Ponder composite policies

Ponder composite policies facilitate policy management in large, complex enterprises. They provide the ability to group policies and structure them to reflect organisational structure, preserve the natural way system administrators operate or simply provide reusability of common definitions. This simplifies the task of policy administrators.

Roles provide a semantic grouping of policies with a common subject, generally pertaining to a position within an organisation. Ponder roles include most of the functionality of RBAC roles described in Section 3, but include obligations. Specifying organizational policies for human managers in terms of manager positions rather than persons permits the assignment of a new person to the manager position without re-specifying the policies referring to the duties and authorizations of that position. A role can also specify the policies that apply to an automated component acting as a subject in the system e.g. a security manager agent. Organisational positions can be represented as domains and we consider a role to be the set of authorisation, obligation, refrain and delegation policies with the *subject domain* of the role as their subject. A role is just a group of policies in which all the policies have the same subject, which is defined implicitly, as shown below.

```

type role ServiceEngineer (CallSDB callSdb) {
  inst oblig serviceComplaint {
    on
    do
      customerComplaint(mobileNo) ;
      t.checkSubscriberInfo(mobileNo, userid) ->
      t.checkPhoneCallList(mobileNo) ->
      investigate_complaint(userid);
    target t = callSdb ; // calls register }

  inst oblig deactivateAccount { . . . }
  inst auth serviceActionsAuth { . . . }
  // other policies
}

```

The role type ServiceEngineer models a service engineer role in a mobile telecommunications service. A service engineer is responsible for responding to customer complaints and service requests. The role type is parameterised with the calls database, a database of subscribers in the system and their calls. The obligation policy serviceComplaint is triggered by a customerComplaint event with the mobile number of the customer given as an event attribute. On this event, the subject of the role must execute a sequence of actions on the calls-database in order check the information of the subscriber whose mobile-number was passed in through the complaint event, check the phone list and then investigate the complaint. Note that the obligation policy does not specify a subject as all policies within the role have the same implicit subject.

Managers acting in organisational positions (roles) interact with each other. A *relationship* groups the policies defining the rights and duties of roles towards each other. It can also include policies related to resources that are shared by the roles within the relationship. It thus provides an abstraction for defining policies that are not the roles themselves but are part of the interaction between the roles. The syntax of a relationship is very similar to that of a role but a relationship can include definitions of the roles participating in the relationship. However roles cannot have nested role definitions. Participating roles can also be defined as parameters within a relationship type definition as shown below.

Many large organisations are structured into units such as branch offices, departments, and hospital wards, which have a similar configuration of roles and policies. Ponder supports the notion of *management structures* to define a configuration in terms of instances of roles, relationships and nested management structures relating to organisational units. For example a management structure *type* would be used to define a branch in a bank or a department in a university and then *instantiated* for particular branches or departments. A management structure is thus a composite policy containing the definition of roles, relationships and other nested management structures as well as instances of these composite policies, has some similarity to an Enterprise Community mentioned in Section 3.

```

type rel ReportingT (ProjectManagerT pm, SecretaryT secr) {
  inst oblig reportWeekly {
    on timer.day ("monday") ;
    subject secr ;
    target pm ;
    do mailReport() ;
  }
  // . . . other policies
}

```

The ReportingT relationship type is specified between a ProjectManager role type and a Secretary role type. The obligation policy reportWeekly specifies that the subject of the SecretaryT role must mail a report to the subject of the ProjectManagerT role every Monday. The use of roles in place of subjects and targets implicitly refers to the subject of the corresponding role.

Figure 7 shows a simple management structure for a software development company consisting of a project manager, software developers and a project contact secretary. [39] gives the definition of the structure.

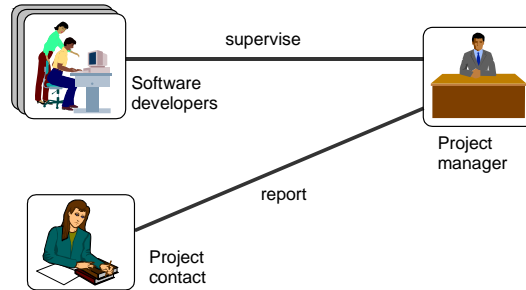


Figure 7: Simple management structure

```

type mstruct BranchT (...) {
  inst role projectManager = ProjectManagerT(...);
  role projectContact = SecretaryT(...);
  role softDeveloper = SoftDeveloperT(...);

  inst rel supervise = SupervisionT (projectManager, softDeveloper);
  rel report = ReportingT (projectContact, projectManager);
}
inst mstruct branchA = BranchT(...);
mstruct branchB = BranchT(...);

```

This declares instances of the 3 roles shown in Figure 7. Two relationships govern the interactions between these roles. A supervise relationship between the softDeveloper and the projectManager, and a reporting relationship between the ProjectContact and the projectManager. Two instances of the BranchT type are created for branches within the organisation that exhibit the same role-relationship requirements.

Ponder allows specialisation of policy types, through inheritance. When a type extends another, it inherits all of its elements, adds new elements and overrides elements with the same name. This is particularly useful for specialisation of composite policies. For example it would be possible to define a new type of mobile systems project manager, from a project manager role with additional policies.

In Ponder a person can be assigned to multiple roles but rights from one role cannot be used to perform actions relating to another role. A person can also have policies that pertain to him/her as an individual and have nothing to do with any roles. In RBAC inheritance is based on policy instances and all policies are defined in terms of roles. This means RBAC requires a much more complicated role structure to separate the policies that are inherited from those that are private.

A compiler has been implemented for the Ponder language. Various back ends have also been implemented to generate firewall rules, Windows access control templates, Java security policies [37] and Java obligation policy rules for interpretation by a policy agent. We also have a system to automatically disseminate policies to the relevant agents that will interpret them i.e. to subjects for obligation and refrain policies and access control agents for authorisation and filter policies.

8. Policy Analysis and Refinement

Despite significant efforts in developing different policy specification techniques, there remain a number of issues to be addressed. In particular, since one of the objectives of using policy-based systems is to fulfil organisational goals, the ability to refine such goals into concrete policy specifications would be useful. As we have discussed in this chapter, it is desirable to maintain the properties of correctness, consistency and minimality when performing any refinement transformation. Of course, this is only possible if the chosen policy specification technique provides support for checking whether these properties hold. To this end, we have developed a mapping of policy specifications into a more formal, logic representation which is based on the Event Calculus [60]. In order to exploit the properties of decidability and lower computational complexity, it is intended that the formal representation would be based on first-order stratified logic. This notation is then used in conjunction with the goal elaboration approach, developed by [42], to develop a usable policy refinement technique.

As part of solving the policy refinement problem, it will be necessary to address some of the outstanding issues related to policy analysis and conflict detection. Unless we have a means of checking for conflicts in a policy specification, it will be impossible to maintain the required consistency property in a given refinement. In Ponder modality conflicts arise between positive and negative policies that apply to the same subjects, targets and actions [64]. These can be detected by syntactic analysis of the policies as the conflict can be determined by detecting overlap of subjects, targets and actions. However, the analysis detects only potential conflicts rather than actual conflicts since constraints may limit the applicability of the policy to disjoint sets of circumstances e.g., different times of day. While modality conflicts can be detected syntactically, other conflicts can only be determined by understanding the actions being performed by the policies. For example, there will be a conflict between two policies that result in the same packet being placed on 2 different queues. Similarly, separation of duty conflicts arise from authorisation policies, which permit the same person to approve payments and sign cheques. Generally, these conflicts are application specific and to detect them it is necessary to specify the conditions that result in conflict. The approach is therefore to specify constraints on the set of policies (i.e. meta-policies) using a suitable notation and then analyse the policy set against these constraints to determine if there are any conflicts [64]. Whether conflicts occur or not may depend on run-time parameters specified in constraints such as time or the current state of the components to which the policy apply. It is thus rather difficult to determine all possible conflicting conditions in advance and so it is still necessary to detect conflicts at run-time. Furthermore, when conflicting policies are detected it is not obvious how to resolve the conflicts automatically. Explicit priority may work in some cases. In some situations, negative authorisation policies should override positive ones, but in other situations the positive authorisation is an exception to a more general negative authorisation. In some situations more specific policies that apply to a department may override general policies applying to the whole organisation. We have been experimenting with meta-policies that define application specific precedence relationships between conflicting policies.

Although some progress has been made in dealing with policy conflicts [3, 64, 102], significant challenges remain to be addressed. In particular, how can one detect conflicts when arbitrary conditions restrict the applicability of the policies? Sometimes, it is possible to compare restrictions placed by the constraints. For example, it is possible to detect if two time intervals overlap or if the policies apply when subjects are in different states e.g., active or standby. Other challenges concern the different levels of abstraction at which policy is specified. Conflicts between organisational goals will inevitably lead to conflicts between the policies derived from these goals. Some policies will trigger complex management procedures, which require the execution of actions that may be specified as part of different policies. This renders the task of ensuring the consistency of a policy specification much more complex.

Recent efforts on formalising the specification of policies and the behaviour of managed systems have gone some way to addressing the challenges of policy analysis and refinement [9]. This work proposes a

formalism that is based on the standard Event Calculus [60] that models both authorisation and management policy specifications together with the behaviour of the managed system. Event Calculus was chosen as both the policies and the management behaviour we are modelling are event driven. Additionally, since an Event Calculus specification of a system can be generated from a state transition model, users can specify the management behaviour using a familiar high-level notation. In a similar fashion, it is possible to translate policies specified in a high-level policy specification language, like Ponder [39], into an Event Calculus representation that describes the semantics of the policy language. This eliminates the need for the user to become conversant with the details of logic programming and the Event Calculus notation.

In order to analyse the policy specification, the Event Calculus representation supports the specification of rules for detecting a range of consistency properties. This includes modality and application specific conflicts such as conflicts of duty together with policy validation. Additionally, the formal representation supports a number of types of review query, allowing the administrator to obtain different views of the information in the policy specification.

This analysis technique uses a combination of deductive and abductive reasoning. Deductive techniques are primarily used for policy validation and to perform review queries whereas abduction is used to detect conflicts between policies. By using abductive reasoning techniques, it is possible to analyse the policy specifications to identify existing conflicts and provide explanations on how they might arise. Because the abduction process is applied to a specification that models both the systems behaviour and the policy specification it is possible to detect conflicts when the applicability of the policies is constrained on the runtime state of the system. Furthermore, by using abduction, the analysis can be performed even with partial specifications of the system state. Having detected conflicts, it is also necessary to have mechanisms for resolving them. Statically defined resolution techniques such as negative policy overrides, and priority assignments have been suggested in the literature [3, 64]. Such approaches suffer from problems of managing consistency in priority assignments and lack of flexibility, making them difficult to use in practice. In order to address this problem, recent research has investigated the application of preference logics for expressing conflict resolution policies in the context of firewall rules [58]. Whilst showing some promise, this approach requires further study before it can be adopted in policy-based management systems.

The Event Calculus formalisation of policy-managed systems can be combined with the KAOS goal elaboration technique [42], to provide a framework for policy refinement. However, the low-level goals derived using this technique cannot be directly used in policies without first identifying the management operations that will achieve them. This identification process is supported by the introducing the concept of a *strategy*. A strategy is the mechanism, by which a given system can achieve a particular goal, i.e., a strategy is the relationship between the system description and the goal. By having a formal specification of the latter two types of information abductive reasoning is used to infer the strategy. The KAOS goal elaboration technique is used because it provides the concept of domain-specific and domain-independent refinement patterns, logically proven goal refinement templates that can be easily reused. Such patterns capture the refinement of goals that are commonly encountered in policy-based management, thus simplifying the refinement process for the user. Additionally the refined policies derived using this process can be encoded in policy refinement patterns that can be later reused when the administrator wishes to satisfy a similar goal. This approach has been applied to refining policies in the context of network QoS management [10].

9. Research Issues

Policies can be used to support adaptability within many different administrative domains. For example a distributed video conferencing application between cooperating organisations in multiple countries may require application specific policies to be implemented within multiple network service providers. There is thus a need to define representations and techniques from transferring policy between the organisations. However the problem is more complex than just one of policy representation. The policies may require knowledge of network topology but an application will not have this information and even one service network provider will not have this information about another service provider. A similar problem occurs when an application or service needs to adapt to changes in an underlying service which it is using. There is

thus a need to negotiate a common policy at a fairly high level of abstraction, i.e. common goals which have predefined mappings to local policies, as it would be unrealistic to expect goals to be automatically mapped into implementable policies. Very little work has been done on policy negotiation but it is possible that some of the concepts from agent plan negotiation may be applicable.

A policy based management system may itself need to adapt to changes in the system being managed due to failures, new situations arising or new application requirements arising. An example would be a security management system which adapts the current policies when it detects that an attack is in progress. This may require policies which dynamically modify the current policies within the system e.g. by activating or deactivating installed policies or selecting new policies to install in the system, thus the target objects on which policies act are other policies and policy interpreters. The next stage in this type of policy feedback loop would be to have a form of adaptive policy which ‘learns’ the best strategy to use, but this is very far off at present.

The whole area of ubiquitous and mobile computing introduces many issues of policy related to security and privacy, particularly when these systems are able to track a user’s activity and location. Techniques are needed for users to be able to specify the conditions under which their identity may be known and what credentials should be revealed for authentication. However in many circumstances users may not wish their current location and activity to be monitored by a ubiquitous computing environment, although these requirements may change in emergency situations. Ubiquitous computing environments provide many new applications for obligation policies. For example how to deal with messages or information being sent to you based on current context – whether it should be forwarded to someone else to deal with, transformed from text to voice or voice to text, what information should be filtered etc. and intelligent environments could detect activity and perform actions on your behalf to control the temperature, switch on appliances or generate warning messages about dangers.

The main problem with the current tools and techniques for specifying policy, whether logic based or otherwise, is that they require considerable computing expertise. None of them are really designed for non-computing literate people to use in order to define the policies they require with respect to security or how to interact with their ubiquitous environment. Even if policy programmers predefine groups of policies for specific roles or situations, there will be a need to adapt these generic templates to specific requirements or to be able to select from a range of those available. It is only recently that research efforts have been directed towards addressing the issues related to these human-centric views of policy specification [16]. However, there is still considerable work to be done in this area.

10. Summary

In this chapter we have presented an overview of the available specification approaches for security, management and enterprise collaboration policies. Policy-based approaches to systems management are gaining widespread interest because they allow the separation of the rules that govern the behavioural choices of a system from the functionality provided by that system. This means that it is possible to adapt the behaviour of a system without the need to recode any of the underlying functionality; and changes can be applied without the need to stop and restart the system. Such features allow administrators to manage systems in a very flexible manner and also provide a potential mechanism for realising the goal of autonomic management of systems.

A common theme across the policy specification notations presented here is that they are focussed in a single functional area – routing, access control or management not a combination of them. Broadly speaking, many of the security policy specification languages are based on logic-based approaches whereas management and enterprise collaboration policy specification notations (with some exceptions) adopt a more informal approach. Whilst it is possible to implement access control policies in the form of ECA rules for the reference monitor, this is not a convenient way of represented authorisations and few systems do it in practice. Ponder is one of the few languages that allows both security and management policies to be represented in a declarative form. Additionally, it supports policy templates that can be configured with application specific parameters as needed and meta-policies that can be used to modify the behaviour of a policy at run-time. Ponder also allows policies to be organised according to roles, management structures and groups. Recent work on the Ponder framework has focussed on redesigning the system to allow

policies to be deployed and executed on resource constrained computing platforms, such as the mobile devices typically found pervasive computing environments [45].

At present, authorisations and event-condition-action rules are the predominant paradigms used in policy-based management, although the latter come in slightly different flavours. Implementation platforms have slowly matured in recent years and increasingly work focuses on formal analysis and refinement of policies. As policy-based techniques are increasingly used to provide adaptation in pervasive systems, ad-hoc networks and collaborations across administrative domains, recent work such as the Promise Theory [27] also focuses on the collaborative aspects of adaptive systems.

Although an essential technique for providing adaptation and management, policies are only one side of the story. To manage large-scale systems and increase the degree of automation they must work in conjunction with other techniques such as advanced metrics, utility functions and planning-based approaches. There has been some very initial work on these integrated techniques.

11. References

- [1] M. D. Abrams. *"Renewed Understanding of Access Control Policies."* In Proceedings of 16th National Computer Security Conf., Baltimore, Maryland, U.S.A., 20-23 September 1993 1993.
- [2] D. Agrawal, S. Calo, J. Giles, K.-W. Lee, and D. Verma. *"Policy Management for Networked Systems and Applications."* In Proceedings of 9th IFIP/IEEE International Symposium on Integrated Network Management, Nice, France, IEEE, May 2005.
- [3] D. Agrawal, J. Giles, K.-W. Lee, and J. Lobo. *"Policy Ratification."* In Proceedings of 6th IEEE International Workshop on Policies for Distributed Systems and Networks, Stockholm, Sweden, IEEE, June 2005.
- [4] G.-J. Ahn and R. Sandhu. *"The RSL99 Language for Role-based Separation of Duty Constraints."* In Proceedings of Fourth ACM Workshop on Role-Based Access Control, Fairfax, Virginia, USA, ACM Press, 28-29 October 1999 1999.
- [5] C. E. Alchourron. *"Normative Systems"*. New York, Wien: xvii+208, 1971.
- [6] Allot Communications. *"Policy Based Networking Architecture"*, <http://www.allot.com/media/ExternalLink/policymgmt.pdf>, 2001.
- [7] R. J. Anderson. *"A Security Policy Model for Clinical Information Systems."* In Proceedings of IEEE Symposium on Security and Privacy, Oakland, California, U.S.A., May 6-8, 1996 1996.
- [8] K. R. Apt, H. A. Blair, and A. Walker. *"Towards a Theory of Declarative Knowledge."* Foundations of Deductive Databases. J. Minker. San Mateo, CA, Morgan Kaufmann: 89-148, 1988.
- [9] A. K. Bandara. *"A Formal Approach to Analysis and Refinement of Policies."* Doctoral Thesis, Imperial College London, London, 2005.
- [10] A. K. Bandara, E. C. Lupu, A. Russo, N. Dulay, M. S. Sloman, P. Flegkas, G. Pavlou, and M. Charalambides (2006). *"Policy Refinement for IP Differentiated Services Quality of Service Management."* IEEE e-Transactions in Networks and Systems Management 3(2), 2006.
- [11] S. Barker. *"Security Policy Specification in Logic."* In Proceedings of Int. Conf. on Artificial Intelligence (ICAI00), Las Vegas, Nevada, USA, June 2000.
- [12] S. Barker (2001a) *"Access Control Policies as Logic Programs."* Technical Report: Imperial College of Science, Technology and Medicine, London, 2001a.

- [13] S. Barker and A. Rosenthal. *"Flexible Security Policies in SQL."* In Proceedings of Fifteenth Annual IFIP WG 11.3 Working Conf. on Database and Application Security, Niagara on the Lake, Ontario, Canada, 15-18 July 2001 2001b.
- [14] J. Barkley, R. Kuhn, L. Rosenthal, M. Skall, and A. Cincotta. *"Role-Based Access Control for the Web."* In Proceedings of CALS Expo Int. & 21st Century Commerce 1998: Global Business Solutions for the New Millennium, Long Beach, CA, USA, 26-29 October 1998 1998.
- [15] J. F. Barkley, K. Beznosov, and J. Uppal. *"Supporting Relationships in Access Control Using Role Based Access Control."* In Proceedings of Fourth ACM Workshop on Role-Based Access Control, Fairfax, Virginia, USA, October 28-29, 1999 1999.
- [16] R. Barret. *"People and Policies: Transforming the Human-Computer Partnership."* In Proceedings of 5th IEEE International Workshop on Policies for Distributed Systems and Networks, IBM TJ Watson Research Centre, New York, USA, IEEE Press, June 2004 2004.
- [17] D. E. Bell and L. LaPadula (1973) *"Secure Computer Systems: Mathematical Foundations and Model."* Technical Report: M74-244, MITRE Corporation, Bedford, MA, 1973.
- [18] E. Bertino, P. Bonatti, and E. Ferrari. *"TRBAC: A Temporal Role-Based Access Control Model."* In Proceedings of 5th ACM Workshop of Role-Based Access Control, Berlin, Germany, 26-28 July 2000 2000.
- [19] K. J. Biba (1977) *"Integrity Constraints for Secure Computer Systems."* Technical Report: ESD-TR76 -372, USAF Electronic Systems Division, Bedford, MA, 1977.
- [20] M. Blaze, J. Feigenbaum, J. Ioannidis, and A. D. Keromytis. *"The Role of Trust Management in Distributed Systems Security."* Secure Internet Programming: Security Issues for Mobile and Distributed Objects. New York, NY, USA, Springer-Verlag: 185 - 210, 1999.
- [21] M. Blaze, J. Feigenbaum, and A. Keromytis. *"Keynote: Trust Management for Public-Key Infrastructures."* In Proceedings of Security Protocols Int. Workshop, Cambridge, England, Springer-Verlag LNCS, April 1998 1998.
- [22] H. Bos. *"Application-Specific Policies: Beyond the Domain Boundaries."* In Proceedings of Sixth IFIP/IEEE Int. Symposium on Intergrated Network Management (IM'99), Boston, MA, USA, 24-28 May 1999 1999.
- [23] A. Boswell (1995). *"Specification and Validation of a Security Policy Model."* IEEE Transactions on Software Engineering **21**(2), 1995.
- [24] D. F. C. Brewer and M. J. Nash. *"The Chinese Wall Security Policy."* In Proceedings of IEEE Symposium on Research in Security and Privacy, Oakland, California, USA, IEEE, May 1989.
- [25] M. Burgess (1995). *"A Site Configuration Engine."* USENIX Computing systems **8**(3), 1995.
- [26] M. Burgess. *"Recent Developments in CfEngine."* In Proceedings of Unix NL Conf., The Hague, 2001 2001.
- [27] M. Burgess. *"An Approach to Understanding Policy Based on Autonomy and Voluntary Cooperation."* In Proceedings of 16th IFIP/IEEE Distributed Systems: Operations and Management (DSOM 2005), Barcelona, Spain, Springer-Verlag, October 2005.

- [28] M. Burgess and F. E. Sandnes. "*Predictable Configuration Management in a Randomized scheduling Framework.*" In Proceedings of IEEE/IFIP Workshop on Distributed Systems Operations and Management (DSOM '2001), Nancy, France, 15-17 October 2001 2001.
- [29] M. Casassa Mont, A. Baldwin, and C. Goh (1999) "*POWER Prototype: Towards Integrated Policy-Based Management.*" Technical Report: HP Laboratories Bristol, Bristol, UK, 1999.
- [30] H. Castaneda. "*The Paradoxes of Deontic Logic.*" New Studies in Deontic Logic: Norms, Actions and the Foundations of Ethics. Hingham, MA, Reidel Publishing Company: 37-85, 1981.
- [31] A. Chandra and D. Harel (1985). "*Horn Clause Queries and Generalizations.*" Journal of Logic Programming 2(1): 1-5, 1985.
- [32] F. Chen and R. S. Sandhu. "*Constraints for Role-Based Access Control.*" In Proceedings of First ACM/NIST Role Based Access Control Workshop, Gaithersburg, Maryland, USA, ACM Press, November 1995 1995.
- [33] L. Cholvy and F. Cuppens. "*Analyzing Consistency of Security Policies.*" In Proceedings of IEEE Symposium on Security and Privacy (S&P97), Oakland, CA, IEEE Press, May 1997.
- [34] J. Chomicki, J. Lobo, and S. Naqvi. "*A Logic Programming Approach to Conflict Resolution in Policy Management.*" In Proceedings of 7th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR2000), Breckenridge, Colorado, USA, Morgan Kaufmann, April 2000.
- [35] Cisco. "*CiscoAssure Policy Networking: End-to-End Quality of Service*", Cisco Systems, 1998.
- [36] D. D. Clark and D. R. Wilson. "*A Comparison of Commercial and Military Computer Security Policies.*" In Proceedings of IEEE Symposium on Security and Privacy, 1987 1987.
- [37] A. Corradi, R. Montanari, E. Lupu, M. Sloman, and C. Stefanelli. "*A Flexible Access Control Service for Java Mobile Code.*" In Proceedings of Annual Computer Security Applications Conf. (ACSAC 2000), New Orleans, Louisiana, USA, IEEE Press, 11-15 December 2000 2000.
- [38] F. Cuppens and C. Saurel. "*Specifying a Security Policy: A Case Study.*" In Proceedings of Ninth IEEE Computer Security Foundations Workshop, Co. Kerry, Ireland, IEEE Press, 1996.
- [39] N. Damianou, N. Dulay, E. C. Lupu, and M. S. Sloman. "*The Ponder Policy Specification Language.*" In Proceedings of 2nd IEEE International Workshop on Policies for Distributed Systems and Networks, Bristol, UK, Springer-Verlag, January 2001.
- [40] DAML. "*The DARPA Agent Markup Language*". <http://www.daml.org>, 2000.
- [41] E. Dantsin, T. Eiter, G. Gottlob, and A. Voronkov. "*Complexity and Expressive Power of Logic Programming.*" In Proceedings of 12th Annual IEEE Conf. on Computational Complexity (CCC'97), Ulm, Germany, IEEE Press, June 1997.
- [42] R. Darimont and A. van Lamsweerde (1996). "*Formal Refinement Patterns for Goal-Driven Requirements Elaboration.*" 4th ACM Symposium on the Foundations of Software Engineering (FSE4): 179-190, 1996.
- [43] DMTF. "*Common Information Model (CIM) Specification, version 2.2*", 1999a.
- [44] DMTF. "*Specification for the Representation of CIM in XML, version 2.0*", 1999b.

- [45] N. Dulay, S. Heeps, E. C. Lupu, R. Mathur, O. Sharma, M. S. Sloman, and J. Sventek. "AMUSE: Autonomic Management of Ubiquitous e-Health Systems." In Proceedings of UK e-Science All Hands Meeting 2005, 2005.
- [46] FIPA. "Domains and Policies Specification", <http://www.fipa.org/specs/fipa00089/>, 2001.
- [47] J. Glasgow, G. Macewen, and P. Panangaden (1992). "A logic for reasoning about security." ACM Transactions on Computer Systems (TOCS) **10**(3): 226-264, 1992.
- [48] T. Grandison. "Trust Management for Internet Applications." Doctoral Thesis, Imperial College London, London, 2003.
- [49] T. Grandison and M. Sloman (2000). "A Survey of Trust in Internet Applications." IEEE Communications Surveys and Tutorials **3**(4), 2000.
- [50] R. J. Hayton, J. M. Bacon, and K. Moody. "Access Control in an Open Distributed Environment." In Proceedings of IEEE Symposium on Security and Privacy, Oakland, California, U.S.A., May 1998 1998.
- [51] A. Herzberg, Y. Mass, J. Michaeli, D. Naor, and Y. Ravid. "Access Control Meets Public Key Infrastructure, or: Assigning Roles to Strangers." In Proceedings of IEEE Symposium on Security and Privacy, Oakland, California, USA, 14-17 May 2000 2000.
- [52] ISO/IEC. "Information Technology - Open Distributed Processing Reference Model - Enterprise Viewpoint", 1999.
- [53] G. Jager and R. F. Stark (1993). "The Defining Power of Stratified and Hierarchical Logic Programs." Journal of Logic Programming **15**(1 & 2): 55-77, 1993.
- [54] S. Jajodia, P. Samarati, M. L. Sapino, and V. S. Subrahmanian (2000). "Flexible Support for Multiple Access Control Policies." ACM Transactions on Database Systems **26**(2): 214-260, 2000.
- [55] S. Jajodia, P. Samarati, and V. S. Subrahmanian. "A Logical Language for Expressing Authorisations." In Proceedings of IEEE Symposium on Security and Privacy, Oakland, USA, IEEE, 1997a.
- [56] A. J. I. Jones and M. J. Sergot (1995). "A Formal Characterisation of Institutionalised Power." Logic Journal of the IGPL **4**(3): 429-445, 1995.
- [57] L. Kagal (2002) "Rei: A Policy Language for the Me-Centric Project." Technical Report: HPL-2002-270, HP Laboratories Palo Alto, CA, USA, 2002.
- [58] A. Kakas, A. K. Bandara, A. Russo, E. C. Lupu, M. S. Sloman, and N. Dulay (2005) "Reasoning Techniques for Analysis and Refinement of Policies for Service Management." Technical Report: DoC # 2005/7, Imperial College London, London, 2005.
- [59] M. Kohli and J. Lobo. "Policy Based Management of Telecommunication Networks." In Proceedings of Policy Workshop 1999, HP Labs, Bristol, UK, 15-17 November 1999 1999.
- [60] R. A. Kowalski and M. J. Sergot (1986). "A logic-based calculus of events." New Generation Computing **4**: 67-95, 1986.
- [61] M. Lee. "Event and Rule Services for Achieving a Web-based Knowledge Network." Doctoral Thesis, University of Florida, 2000.

- [62] J. Lobo, R. Bhatia, and S. Naqvi. "A Policy Description Language." In Proceedings of 16th National Conf. on Artificial Intelligence, Orlando, Florida, USA, 18-22 July 1999 1999.
- [63] E. C. Lupu and M. S. Sloman (1997b). "Towards a Role Based Framework for Distributed Systems Management." Journal of Network and Systems Management **5**(1): 5-30, 1997b.
- [64] E. C. Lupu and M. S. Sloman (1999). "Conflicts in Policy-Based Distributed Systems Management." In IEEE Transactions on Software Engineering - Special Issue on Inconsistency Management **25**(6): 852-869, 1999.
- [65] M. Martinez, M. Brunner, J. Quittek, F. Strauß, J. Schönwälder, S. Mertens, and T. Klie. "Using the Script MIB for Policy-based Configuration Management." In Proceedings of IEEE/IFIP Network Operations and Management Symposium (NOMS 2002), Florence, Italy, IEEE, April 2002.
- [66] N. H. Minsky (1991). "The Imposition of Protocols Over Open Distributed Systems." IEEE Transactions on Software Engineering **17**(2): 183-195, 1991.
- [67] N. H. Minsky and P. Pal. "Law-Governed Regularities in Object Systems - Part 2: A Concrete Implementation." Theory and Practice of Object Systems (TAPOS), John Wiley. **2**, 1997.
- [68] J. Moffett. "Control Principles and Role Hierarchies." In Proceedings of Third ACM/NIST Role Based Access Control Workshop, Fairfax, Virginia, USA, ACM Press, 22-23 October 1998 1998.
- [69] B. Moore, E. Ellesson, J. Strassner, and A. Westerinen (2001). "Policy Core Information Model -- Version 1 Specification." Network Working Group - RFC3060, <http://www.ietf.org/rfc/rfc3060.txt>, 2001.
- [70] OASIS. "XACML language proposal, version 0.8", 2001.
- [71] OMG. "Object Constraint Language Specification, version 1.3", 1999b.
- [72] R. Ortalo. "A Flexible Method for Information System Security Policy Specification." In Proceedings of 5th European Symposium on Research in Computer Security (ESORICS 98), Louvain-la-Neuve, Belgium, Springer-Verlag, September 1998 1998.
- [73] OWL. "The Ontology Web Language". <http://www.w3.org/TR/owl-features/>, 2004.
- [74] H. Prakken and M. J. Sergot. "Dyadic Deontic Logic and Contrary-to-duty Obligations." Defeasible Deontic Logic: Essays in Nonmonotonic Normative Reasoning. D. Nute. Boston, Kluwer Academic Publishers. **Synthese Library: 263**: 223-262, 1997.
- [75] C. Ribeiro, A. Zuquete, and P. Ferreira. "SPL: An access control language for security policies with complex constraints." In Proceedings of Network and Distributed System Security Symposium (NDSS'01), San Diego, California, February 2001 2001a.
- [76] C. Ribeiro, A. Zuquete, and P. Ferreira. "Enforcing Obligation with Security Monitors." In Proceedings of Third Int. Conf. on Information and Communications Security (ICICS 2001), Xian, China, 13-16 November 2001 2001b.
- [77] P. Samarati and S. Vimercati. "Access Control: Policies, Models, and Mechanisms." Foundations of Security Analysis and Design (Tutorial Lectures). R. Focardi and R. Gorrieri, Springer -Verlag: 137-196, 2000.

- [78] R. Sandhu, D. Ferraiolo, and R. Kuhn. "*The NIST Model for Role-Based Access Control: Towards A Unified Standard.*" In Proceedings of 5th ACM Workshop on Role-Based Access Control, Berlin, Germany, 26-28 July 2000 2000.
- [79] R. Sandhu and P. Samarati (1994). "*Access Control: Principles and Practice.*" IEEE Communications Magazine **32**(9): 40-48, 1994.
- [80] R. S. Sandhu. "*Role Activation Hierarchies.*" In Proceedings of Third ACM/NIST Role Based Access Control Workshop, Fairfax, Virginia, USA, ACM Press, 22-23 October 1998 1998.
- [81] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman (1996). "*Role-Based Access Control Models.*" IEEE Computer **29**(2): 38-47, 1996.
- [82] J. R. Searle (1969). "*Speech Acts: An Essay in the Philosophy of Language*". Cambridge, Cambridge University Press, 1969.
- [83] M. J. Sergot, F. Sadri, R. A. Kowalski, F. Kriwaczek, P. Hammond, and H. T. Cory (1986). "*The British Nationality Act as a logic program.*" Communications of the ACM(29): 370-386, 1986.
- [84] M. S. Sloman (1994b). "*Policy Driven Management for Distributed Systems.*" Journal of Network and Systems Management **2**(4): 333-360, 1994b.
- [85] Y. Snir, Y. Ramberg, J. Strassner, R. Cohen, and B. Moore. "*Policy QoS Information Model*", 2001.
- [86] J. M. Spivey (1989). "*An Introduction to Z and Formal Specifications.*" IEE/BCS Software Engineering Journal **4**(1): 40-50, 1989.
- [87] M. W. A. Steen and J. Derrick. "*Formalising ODP Enterprise Policies.*" In Proceedings of 3rd Int. Enterprise Distributed Object Computing Conf. (EDOC '99), University of Mannheim, Germany, IEEE Publishing, September 1999 1999.
- [88] M. W. A. Steen and J. Derrick (2000). "*ODP Enterprise Viewpoint Specification.*" Computer Standards and Interfaces **22**: 65-189, 2000.
- [89] G. N. Stone, B. Lundy, and G. G. Xie (2001). "*Network Policy Languages: A Survey and a New Approach.*" IEEE Network: 10-20, 2001.
- [90] J. Strassner (2004). "*Policy-based Network Management*". San Francisco, Morgan Kaufmann, 2004.
- [91] J. Strassner and E. Ellessen. "*Terminology for describing network policy and services (version 00)*", 1998.
- [92] J. Strassner, E. Ellessen, B. Moore, and R. Moats. "*Policy Core LDAP Schema*", 2002.
- [93] S. Y. W. Su, H. Lam, M. Lee, S. Bai, and Z.-J. M. Shen. "*An Information Infrastructure and E-services for Supporting Internet-based Scalable E-business Enterprises.*" In Proceedings of 5th IEEE Annual Enterprise Distributed Object Conf. (EDOC2001), Seattle, WA, IEEE Computer Society, September 2001.
- [94] D. L. Tennenhouse, J. M. Smith, W. D. Sincoskie, D. J. Wetherall, and G. J. Minden (1997). "*A Survey of Active Network Research.*" IEEE Communications Magazine **35**(1): 80-86, 1997.
- [95] R. K. Thomas. "*Team-based Access Control (TMAC): A Primitive for Applying Role-based Access Controls in Collaborative Environments.*" In Proceedings of Second ACM/NIST Role Based Access Control Workshop, Fairfax, Virginia, USA, ACM Press, November 1997 1997.

- [96] D. Thomsen, D. O'Brien, and J. Bogle. *"Role Based Access Control Framework for Network Enterprises."* In Proceedings of 14th Annual Computer Security Applications Conf., December 1998 1998.
- [97] G. Tonti, J. Bradshaw, R. Jeffers, R. Montanari, N. S. Suri, and A. Uszok. *"Semantic Web Languages for Policy Representation and Reasoning: A Comparison of KAoS, Rei, and Ponder."* In Proceedings of 2nd International Semantic Web Conference, Florida, USA, Springer-Verlag, 2003.
- [98] V. Ungureanu and N. H. Minsky. *"Unified Support for Heterogeneous Security Policies in Distributed Systems."* In Proceedings of 7th USENIX Security Symposium, San Antonio, Texas, January 1998.
- [99] V. Ungureanu and N. H. Minsky. *"Establishing Business Rules for Inter-Enterprise Electronic Commerce."* In Proceedings of 14th Int. Symposium on Distributed Computing (DISC 2000), Toledo, Spain, Springer-Verlag, October 2000.
- [100] A. Uszok, J. Bradshaw, R. Jeffers, N. Suri, P. Hayes, M. Breedy, L. Bunch, M. Johnson, S. Kulkarni, and J. Lott. *"KAoS Policy and Domain Services: Toward a Description-Logic Approach to Policy Representatin, Deconfliction and Enforcement."* In Proceedings of 4th IEEE Workshop on Policies for Networks and Distributed Systems (Policy 2003), Lake Como, Italy, IEEE, June 2003.
- [101] D. Verma, M. Beigi, and R. Jennings. *"Policy Based SLA Management in Enterprise Networks."* In Proceedings of Policy Workshop 2001, HP Labs, Bristol, UK, Springer-Verlag, 29-31 January 2001 2001.
- [102] D. C. Verma (2001). *"Policy-Based Networking: Architecture and Algorithms"*, New Riders Publishing, 2001.
- [103] A. Virmani, J. Lobo, and M. Kohli. *"Netmon: network management for the SARAS softswitch."* In Proceedings of 2000 IEEE/IFIP Network Operations and Management Seminar (NOMS 2000), Hawaii, April 2000 2000.
- [104] G. H. von Wright (1951). *"Deontic Logic."* Mind **60**: 1-15, 1951.
- [105] R. J. Wieringa and J.-J. C. Meyer. *"Applications of Deontic Logic in Computer Science: A Concise Overview."* In Proceedings of Practical Reasoning and Rationality (PRR 98), Brighton, UK, John Wiley & Sons, August 1998.

On the Complexity of Change and Configuration Management

Mark Burgess¹
Lars Kristiansen^{1,2}

December 12, 2006

¹ Oslo University College, Faculty of Engineering
and

² Department of Mathematics, University of Oslo

Contents

1	Introduction	2
1.1	Configuration and systems	3
1.2	How to get there from here	4
1.3	Static or dynamic configuration?	4
1.4	Goals, Policies and Behaviours	5
2	Configuration operations as an automated form of System Administration	6
2.1	The fundamental hypothesis	6
2.2	Configuration defined	8
2.3	The importance of coding	10
2.4	Some examples	12
2.4.1	Web Server Configuration	12
2.4.2	Example: A trivial fragment in \mathbb{B}^3	13
2.4.3	File Permissions	14
2.5	Implementability and Maintainability	14
2.6	The algorithmic complexity of configuration management	16
3	A gentle introduction to complexity theory	16
3.1	What is a ‘problem’?	17
3.2	The class P	17
3.3	Proof techniques	19
3.4	A hard problem	21
4	Analysis of some configuration problems	23
4.1	The complexity of the system coding	23
4.2	Operators formalised	24
4.3	Some problems on operators: IDM , INJ and CON	26
4.4	The Complexity of IDM and INJ	27

5	More complexity theory	31
5.1	The space classes LINS _{SPACE} and PSPACE	31
5.2	Example: SAT is in LINS _{SPACE}	32
5.3	Non-Determinism and the class NP	34
5.4	Example: SPLIT and SAT are in NP	36
5.5	The class co-NP	37
5.6	More on the relationship between the complexity classes	37
5.7	The importance of the complexity classes	39
6	More on operators and configuration management	40
6.1	More on the complexity of IDM and INJ	40
6.2	The Complexity of CON and CON_x	43
6.2.1	A brief discussion	43
6.2.2	Turing Machines	44
6.2.3	Operators Simulating Turing Machines	47
6.2.4	CON_x is PSPACE-hard.	48
6.2.5	Our last words on the complexity of CON_x and CON	49
7	Conclusions	49
7.1	A summary	49
7.2	Approximation or bust	51
7.3	Last words	51

1 Introduction

How hard is it to build and tune a computer system so that one can predict its operational behaviour from an original specification?

In computer programming, the boundaries of this problem have been considered at length (see the chapter by Bergstra and Bethke in this volume, for instance). In system administration, the framework for the problem is both larger and potentially much less reliable. System administration is not only a set of procedures executed on a computer, it also involves the interaction of humans in a more more pervasive way. It raises questions of a pilosophical nature: are we really in control of computer behaviour, or are there limitations to what we can achieve? In computer programming, one does not normally take into account the random arrival of input, or the interactions between independent tasks which share common resources. These are matters that are of central importance in system administration.

Suppose we were indeed able to rule or govern the behaviour of a computer system, by choosing the values of all the right registers, how hard would it be to find out what values or patterns of data were required to map the specification of those values to a desired behaviour? How much effort would we need to put in to solve an arbitrarily posed problem of this kind? What are our chances of being able to optimize it, or simply do it at all?

These are questions we discuss in this chapter. To begin formulating an answer we must cross several bridges, from empirical system administration and more formal computational disciplines such as languages and algorithmic complexity. We discuss what it means to talk about the difficulty of arranging a particular computer configuration and why we believe the choice of configuration has something to do with its ultimate behaviour.

There are many measures one might use to discuss the complexity of a solution to a problem: the number of lines of code required to solve the problem, the number of

hours of planning needed, the CPU cycles expended during the task, the memory resources used to run the algorithms, etc. All of these are potential measures of difficulty or complexity, but to answer the question in a comprehensible way, we have to pose the question in a way that can be analysed.

None of the measures above have any meaning to system administration until we can identify a set of acceptable tests and determine the answer to a question: does the solution pass the test? Yes or no? So, how shall we proceed? Our story is a complicated one; it requires both a broad background and lengthy argumentation. To make it comprehensible to the widest possible audience, we have chosen to use a pedagogical style with a moderate amount of introductory background. We hope that this makes our discussion as self-contained as possible. Perhaps half of the material will be known to about half our audience, but we doubt that these halves will overlap.

We base our discussion on classical computational complexity theory[1, 2] and the modern theory of configuration management in terms of operations[3, 4, 5]. Many readers will have heard of phrases like “NP complete” used to describe hard problems, and understand that therein lie dæmons and dragons concerning the difficulty of a task. But what does this really mean? Similarly, readers will have heard about the problem of configuration management and computer maintenance. Why is this any different from computer programming? We begin from the context of computer management.

1.1 Configuration and systems

The problem of configuring (and sometimes maintaining) a computer system from an original specification has been discussed in the literature in a variety of settings[6, 3, 7, 4, 8, 9, 5]. Much software has been written over the years purporting to solve the problem in some manner[10, 11, 12, 13, 14].

We understand intuitively that a *configuration* is an assemblage of components and parts that makes up a complete system. The term appears both in system administration and in software engineering with analogous meanings. According to the IEEE software engineering glossary[15], “Configuration Management is the process of identifying and defining the items in the system, controlling the change of these items throughout their life-cycle, recording and reporting the status of items and change requests, and verifying the completeness and correctness of items.”

Although the goal of ‘configuration management’ in software engineering is somewhat different than in system administration, this definition still fits quite well the general idea for system administration too, but let’s be more specific.

Suppose we start with a blank computer and install an operating system from source. We are now able to switch on the computer and give it instructions, but this is rarely enough to make it into a useable device within some organization or home. In addition, we must add software, set a number of defaults and preferences such as how to print and how to access information from a network, and then we must consider what users may log in, share data, what permissions resources must have, which services we should run from the host or make use of from outside.

In biology, this is analogous to the procedure in which generic stem cells are differentiated into dedicated bodily cells in a growing foetus. In other words, what will the purpose of the computer be? And how should its running state be managed so that it will fulfill that purpose for the time we require and the specific (or unspecific) list of users who require it? The task of configuration management is to turn our expectations of behaviour into a number of operations on the working computer

such that it will fulfill our needs.

This all sounds eminently reasonable, but can it really be done? Of course, arranging the pieces of a puzzle is a straightforward task of writing data to a disk or memory device, so what's the problem? The problem, it turns out, is not so much implementing these patterns themselves, but in finding them and in deciding what they mean.

1.2 How to get there from here

Suppose we know what policy, configuration, state, or pattern of data, we want to implement on a computer, i.e. we have solved a problem and found a specification that we want to implement. We must know translate this desired state into a set of *configuration operations* (or operators). There are then two ways of proceeding with the operations we need for configuration implementation. They are roughly analogous to procedural programming and logic programming[16].

- We specify the starting point, e.g. a blank computer, and we specify a sequential (imperative) programme of steps to transform it into the desired state.
- We specify the final state and leave it up to the (declarative) program to compute any suitable algorithm for getting there.

In neither case is it guaranteed that we shall be able to reach the configuration state we had in mind at the start, but let us set aside that problem for now.

Consider an example. If we fix a bit string, like a file mode, using a numerical value, there is little ambiguity in the procedure. It seems like a straightforward problem to write some configuration to a computing device (e.g. `chmod 755 filename`). It is straightforward, easy, uncomplex. There is little difference between the two alternatives for such a trivial change.

But now consider the expression of a high level behaviour, e.g. `InstallPackage(webserver)`. This is no longer straightforward, because it describes a configuration change only at a medium level, not all the way down to the bits. This is like saying: "Make me prettier!" It is not a uniquely defined or reproducible goal. We therefore need to define this at a low level – and now it does matter which of these two approaches we choose. Someone might tell you that it is their goal to make you prettier, but you cannot guarantee their behaviour from this assertion. You might trust them more, if they told you about what end result they were going to guarantee.

An imperative beautician might specify: first I will put blue on your eyes, then I will put red on your lips, and finally I will take a centimetre off your hair. The declarative beautician might say: I will make sure that your eyelids are blue and your lips are red and that your hair is 10cm long. In the first case, the operation could go drastically wrong if the configuration object was wearing sun-glasses and only has 2cm long hair to begin with.

The problem is that, if we base our actions on assumed *pre-conditions*, rather than on desired *post-conditions*, procedures are fragile and one needs to be certain about that initial configuration. See the chapter by Sun and Couch in this volume for a detailed discussion of the imperative approach.

1.3 Static or dynamic configuration?

There is another more important complication to this simple viewpoint: the view described above is too static. A computer system is not static like a picture or

painting, it is more like a board-game with players who are changing the state of a board at all times. The configuration problem described above is only analogous to setting out the pieces at time $t = 0$. Once we let the computer run, interact with users and change its operational state, the pieces move and the configuration changes[6]. Do we want them to move? Who are we playing against? Is it the system versus the users, or is it a maintenance agent versus gremlins? We need to understand why the system is changing and how much of it we want to keep constant.

In fact, we do not necessarily care about the exact pattern of every byte and register on the computer – only certain patterns will affect the behaviour that we are interested in, others will have no effect on it. For instance, if we decide to run a Web service that responds withing a maximum of 5 milliseconds, we do not need to specify the contents of every text file and picture file on the computer in order to achieve that goal. There will be a few key files that affect the functionality; all other files are irrelevant (symmetrical with respect to the goal of making a Web service). Indeed, our specification of the behavioural configuration does not usually extend to the *data content* of user files and images – that is determined by a different set of policies, specifications or goals.

There is thus a *separation of concerns*, between different parties and their interests in the state of the system which complicates the matter in system administration. In a computer program, there is one overriding programmer that decides authoritatively over a domain and all of its subdomains. In system administration, the separation is less clear.

Separation of concerns is a common problem in computer science, and it is generally handled by encapsulating independent concerns in independent data structures using some kind of high level coding of the computer's state. Object Orientation and Aspect Orientation are examples of this in computer programming. At the level of system resources and their administration, this is partially achieved using data structures abstractions like *files*, *directories* and *processes*. However, there is no operating system in use today that achieves a clean separation of operational and configuration concerns, or even complete separation between subsystems. This makes system administration more difficult and partially explains the current revival of interest in virtual machine technologies which allow individual services to be isolated.

1.4 Goals, Policies and Behaviours

An important development which changed thinking about configuration management was the notion of policy based management[17, 18, 11] (see the chapter by Bandara et al in this volume)¹.

We understand policy as some supposed catalogue of decisions, whose outcomes have been pre-determined, like a code-book of behaviours. When an expected situation arises, there is already an answer to the problem encoded in policy and we merely have to look up the appropriate behaviour. This idea is quite general and has been applied in many arenas, both for runtime behaviour and configuration choices. A policy can be thought of as a somewhat advanced notion of *preferences*, in which the semantics of certain preferences have been defined in advance.

In configuration management, policy allowed a move away from explaining configu-

¹In fact this was two separate developments at approximately the same time which later came together into a common conceptual framework. The term Policy Based Management is due to Sloman et al.

ration in the imperative terms of an algorithmic recipe of *configuration operations*, towards the specification of declarative guidelines for the end goal itself. This conceptual leap was important because it refocused the attention on final behaviour rather than intermediate state. Policy about the final state is easily associated with post-condition based configuration. The semantics of post-condition policy are easier to fix than an attempt to construct a scenario by imperative recipe.

Of course, by focusing on goal-oriented, post-condition policies one does not remove the need to find an algorithm for implementing the end result. However, this approach furnishes us with a neat framework for discussing the *complexity* of solving a configuration problem. We can make a much more direct attack on the problem of configuration management's algorithmic complexity by studying ways of achieving a fixed final result, satisfying certain properties.

2 Configuration operations as an automated form of System Administration

2.1 The fundamental hypothesis

What does configuration management entail? It is more than simply implementing changes. Picking some arbitrary configuration for a system is easy (one could roll a die or set all values in the system to zero, etc.), but picking a configuration that satisfies a highly particular set of constraints could be a very non-trivial request—and policy, after all, is all about setting constraints on what is allowed or desired.

The often unstated assumption about configuration management is the following (see fig. 1):

Hypothesis 1 (Policy Guided Behaviour) *A high level policy can be used to determine a number of equivalent low level configurations on a computer system. There will then be a direct association between a "correctly configured computer" and a "correctly behaving computer", where "correct" means "policy or specification compliant".*

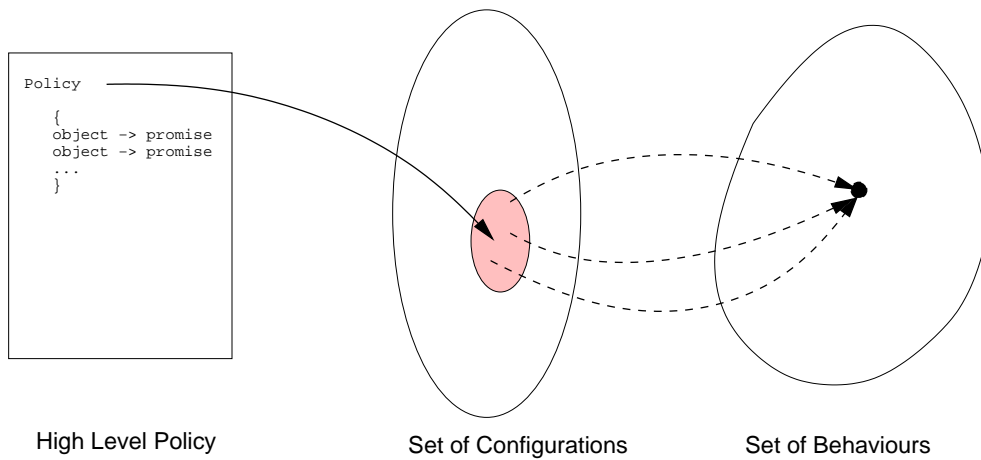


Figure 1: The main hypothesis in configuration management is that there is an association between high level policy, low level configuration and eventual behaviours.

It should be stated straight away that this hypothesis remains to be proven. Most system administrators have a gut feeling that it is approximately true, but there is no sense in which we have a formal proof of this. The reason is that computer systems are not simply deterministic automata. They receive unpredictable input from users through their input/output channels, and the software that executes on the systems is not usually considered to be a part of the configuration itself, thus the behaviour of the system is not fully determined.

In practice what one normally means by this hypothesis is that, for a constant configuration, the system will behave *predictably* on average. However, it is important to note an important objection: it is far from clear that an arbitrary policy will be stable – or by implication, that an arbitrary configuration will be *sustainable* over time, without a sufficiently invasive maintenance process.

An approximate understanding of the hypothesis has been developed in ref. [6], using a post-condition view of configuration. There it was shown that, by limiting the set of definable policies to those that can be faithfully represented and maintained over time, one can associate probable computer behaviour with an initial policy.

$$\text{Policy} \mapsto \frac{\text{Configuration}}{\text{Equivalences}} \mapsto \text{Behaviour} \quad (1)$$

We read this as follows. Any system can be understood as a balance between the freedoms to change and the constraints which limit that change[7]. A policy implies a number of constraints which partially determine a configuration of the system up to certain equivalences (which are implied by the freedoms that policy does not constrain) and it is further assumed that those equivalence lead to equivalent behaviour.

This fulfils the intention of the hypothesis, but it does so by restricting the notion of policy in order to lead to make it true. If we want sustainable behaviour, it seems that it can be achieved for a subset of possible policies for which this property holds, but we do not really know how to find that subset.

There will always be fluctuations in a configuration, due to environmental effects, and these might be relevant or irrelevant to the observed behaviour. Thus the best one can hope to sustain (by correcting policy deviant errors continuously) is a time-average behaviour. In effect, the model likens configuration management to Shannon’s problem of error correction in a noisy channel[19, 20].

An alternative approach to studying the problem has been suggested in ref. [5], based on a pre-conditional model of configuration changes. In that case, there are no such restrictions on policy other than reachability from an initially known state. In that case, it is quite unclear whether there is a predictable behaviour for every possible composition of configuration operations. The task is then to weed out those reproducible behaviours that one would like to see.

The hypothesis has characteristics akin to the general problem of programming semantics (see chapter by Bergstra and Bethke), in which the policy is a program, the configuration is an intermediate code that is then executed on a (virtual) machine. The main difference here is that the execution machine is not necessarily either constant or reliable over the time interval for which policy is expected to be maintained. This is partially due to the presence of interactions with the external environment[6]. Hence we must verify each of the outcomes by explicit measurement to be certain of the persistence of the resulting behavioural promises.

2.2 Configuration defined

We discuss now the essence of configuration management in terms of a specific model, using the idea of patterns and configuration operators (or functions)[3, 7]. The description of discrete patterns is well known in computer science as the theory of formal grammars[21]. Any pattern must be formed from a number of distinct symbols (the set of which is called the *alphabet* Σ of the language), and the specific rules to which a pattern conforms determine its *language* L . Patterns divide the set of languages into four levels of grammatical complexity known as the Chomsky hierarchy[21], which we shall not discuss further in this essay. We then define a general configuration as follows:

Definition 1 (Configuration) *A configuration is a discrete pattern that is exhibited by a computer system. It is a string belonging to some language L , with alphabet Σ .*

In this essay, we shall ultimately confine ourselves to the simplest binary alphabet consisting of the symbols $\Sigma = \{0, 1\}$, from which we can code any other. However, we must acknowledge that there are many levels of coding that are represented in a computer system.

A configuration can, in other words, be thought of as a complicated state: it is an instance of a pattern q found in a specified region of system's resources. We think of any such region as being a system 'variable', such as the contents of a register, a file or an abstract container such as a Process Control Block inside a process dispatcher.

These definitions are clear enough, but they are too abstract to say much about configuration management, as we understand it. What we are really interested in is the ability to describe *changes* of configuration, i.e. the reorganisation of resources into new patterns. To make the discussion fit for analysis, we need a formal representation for that too.

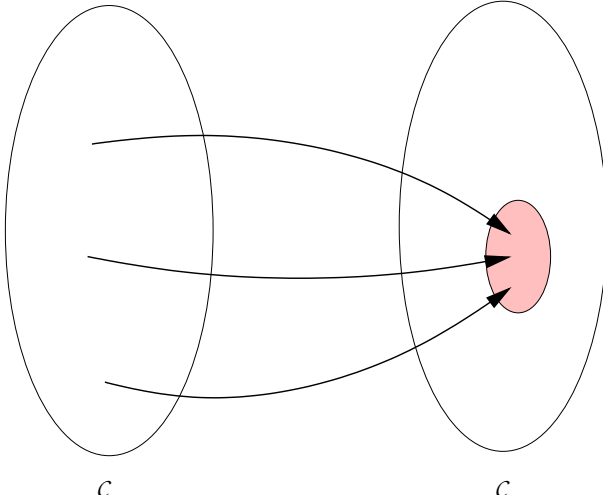


Figure 2: A change in configuration is generally a mapping from some general configurations into a subset of policy-compliant configurations.

A change from one configuration state q to another state q' takes place as a result of an *operation*, and in the current paper we will view as one iteration of a function

or mapping

$$f : \mathcal{C} \rightarrow \mathcal{C} \quad (2)$$

where \mathcal{C} is the set of all possible configurations (see fig. 2). A *policy* is then, according to eqn (1), a specification of some range of configurations that we want to allow[6], and the set of policy compliant configuration \mathcal{P} will be a (normally very small) subset of \mathcal{C} , i.e. $\mathcal{P} \subseteq \mathcal{C}$.

If we want to maintain the equivalence between actual state and the idealized state implied by policy, we need to repair or maintain it (perhaps repeatedly). We can think of a repair strategy as being an operation (or function) f_r which, when repeated a sufficiently number of times, will take the system from an arbitrary configuration to a policy compliant configuration, that is, for all sufficiently large $k \in \mathbb{N}$ we have²

$$f_r^k(x) \in \mathcal{P} \text{ for any } x \in \mathcal{C} \quad (3)$$

Before discussing the details of such (configuration) operations, we must spend a section clearing up the role played by configuration coding, or transformations of alphabet, in the complexity of configuration management.

However, before leaving this section, we must briefly note another assumption that we are making about configuration. We are not being completely clear about the distinction between *configuration* and *state*. The concept of state is fairly well understood in computer science, particularly in relation to the theory of languages and automata. A working computer contains state in every part of its memory, but it would be wrong to think of a real computer as a closed Turing machine that deterministically rewrites the contents of all its possible states according to a fixed plan.

In configuration management, we usually only try to determine a part of the computer's total state – that part which contains the installed software and its customizations, settings and preferences, Such state is recorded in software and in configuration files and databases as options on their behaviours. We do not typically imagine trying to control the internals of different programs' states, or model the operational history of processes that learn from their past experiences, e.g. customer transactions or the contents of databases that we do not believe have any *direct* affect on the behaviour of the system. Such operational state does however enter into the observation of a system, as a reflection of its behaviour.

In a sense, we glibly assume that the state of the system can be divided into two parts: a part that affects behaviour, and a part that is changing, but which has no direct influence on behaviour.

Hypothesis 2 (Separation of state) *A computer's total state can be divided into two parts: configuration state which determines behaviour, and operational state which is merely the result of behaviour.*

As with the first hypothesis, it is not clear that this must always be true. What about the contents of software itself? Software performs operations on the state of the system of a kind that we are not even modelling, despite being not so fundamentally different from configuration operations. Software is, in a real sense, a part of configuration state as it affects behaviour, and yet we do not control its contents or behaviour in detail.

² $f^k(x)$ denotes the result of repeatedly applying the function f , i.e. $f^0(x) = x$ and $f^{n+1}(x) = f^n(f(x))$

The hypothesis above is, in fact, directly analogous to the division one makes in describing the physics of dynamical systems[7]. It is a separation of state into “slow” and “fast” changing variables. What we are really postulating is the existence of two separable *time-scales* – a slow time-scale over which humans can make high level decisions, and a fast time-scale at which the computer performs its own manipulations. We hope that these are separable, else the model breaks down.

The interface between the fast and the slow is software. In practice, we only decide to install or not install certain packages of software, and there is nothing to stop a program from using the contents of fast operational state for affecting and modifying behaviour, which we then try to regulate by making slow changes of configuration. Clearly, we are assuming an idealization in order to make limited progress.

We shall suppress further discussion of this point below and focus only on the supposed slowly-changing *configuration* or *configuration state*. Part of the real problem with configuration management, and the difficulty in fully believing in our first hypothesis derives from the uncertainties described here.

2.3 The importance of coding

System administration is performed at a high level of abstraction, but the aim of configuration management is to eventually map these high level concerns into precise operations on the resources of the computer[22, 6, 7]. By a *resource* we mean either an abstract data object or a hardware capability, represented through its various control registers.

The basic configuration states of a computer system and its resources are therefore coded within its memory. At a fundamental level, the memory takes the form of a finite array of *bits*, of length n whose configuration value at any given moment is a single instance of the set of all such strings \mathbb{B}^n , to which we return presently. This is the *configuration space* of the system. We denote this lowest level representation of the resources as R^1 , in anticipation of higher level codings to follow.

Written onto the configuration space of bits is a pattern of values, zero or one, which changes over time; thus, a configuration may be defined as follows: A level 1 configuration $q_1(x, t)$, on R^1 , is a pattern of Boolean values associated with each element of the configuration space, at a fixed snapshot in time t :

$$q_1 : R^1 \rightarrow \{0, 1\}, \quad (4)$$

Readers can follow this formalization in more detail in ref. [6]. A configuration at some point changes in time as a *chain*, or *process*[23].

Although we understand that computer systems are bit-configurations, operated on by low level rules, the viewpoint ignores an important issue, which is the layered coding of information used to represent user-level data. Most data and processes require quite complicated representations that require one to deal with high level objects such as files (containing sub-languages) and directories. It is at these levels that both the usage and the administration of the system take place, so it is this level to which one must address a model of administration.

We accomplish the transition from low level to high level by successive levels of grouping smaller objects into larger ones, such as the hierarchy of objects expressed in table 1. For simplicity, one can assume that all the objects have the same size. This is not the case in practice, but the assumption serves to avoid unnecessary additional specification and does not alter the argument, only the details.

These items represent effective properties of the system, i.e. new alphabets of non-overlapping objects. At level ℓ , the system may be considered ℓ -dimensional, since

Level	Example objects
6	Groups, departments, LANs
5	Users, virtual machines, agents
4	Compound objects, files and attributes
3	int, float, char etc
2	bytes, words
1	bits (RAM,ROM,disk)

Table 1: A separation of scales in a computer system. At a certain level, human users begin to interact with the system, and thus form part of it.

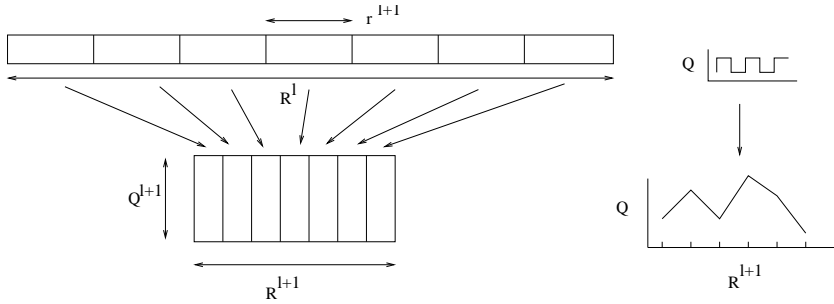


Figure 3: A partitioning of the configuration space into higher level objects, is a mapping from one to two dimensions. The vertical scale is the range of the coding level; as the level increases so does the range of values taken by objects at that level. For the purpose of discussing rates of change, these values can be represented as numerical values.

there are up to ℓ independent degrees of freedom. (A degree of freedom is a capacity for independent change.)

Suppose we make R^ℓ the set of D_ℓ objects of level ℓ , and let Q_ℓ be the set of d_ℓ bits within each element of R^ℓ (see fig. 3). A high level coding of the configuration space R^1 , is then understood simple as a mapping L , which coarse-grains a set of D_ℓ lower level objects into $D_{\ell+1}$ higher level objects, of size $d_{\ell+1}$ [6]:

$$L_\ell : R^\ell \rightarrow R^{\ell+1}. \quad (5)$$

Now, these higher level codings are constructed in much the same way that languages are constructed from grammar rules[24]. For example, most structures can be captured by the notion of a context free grammar $G = (N, \Sigma, P, S)$ where N is an alphabet of non-terminal symbols, Σ is an alphabet of terminal symbols, S is a starting symbol from N and P is a set of production schemas. Each $p \in P$ has the form $n \rightarrow \omega$, where $n \in N$ and $\omega \in (N \cup \Sigma)^*$. Context free grammars can be extended to allow a general regular expression over the total alphabet in place of ω . We shall use this idea later in section 4.1.

A high level configuration is thus no longer viewed in terms of elemental bits, but in terms of high level objects such as numbers, symbols or file objects. The potential range of values represented by such objects increases exponentially in proportion to their size. In practice, the values used by different software systems will be considerably less than this.

This mapping takes bit patterns into a higher alphabet of effective symbols. Thus the set of all possible configurations \mathcal{C} referred to earlier is an ambiguous quantity in terms of symbols, since it depends on the level at which we choose to view it. The only unambiguously general way of discussing it is to view it in terms of its

atomic bits, i.e. to replace \mathcal{C} in the discussion with \mathbb{B} .

It is natural to ask the question: how much of the complexity of configuration management arises from the coding itself? If the coding is the major part of the complexity then the problem is artificially hard. We shall discuss this further below, but one can immediately be sceptical of this idea. If transforming the coding of the system itself were a hard problem, in the sense of algorithmic complexity then it would never have been implemented on computers, as it would take too long. But these computers exist and work more or less well. Thus it seems unlikely that the coding could have an important bearing on the matter. We return to this in section 4.1.

2.4 Some examples

Notation can be an important cultural bridge. In this essay, we work with two different notations so that readers coming from different backgrounds will recognise the text. In the definitions below, we describe an operator expression often as a tuple which gives the output as a number of expressions that show how the input is combined to yield the output. We also write this as a vector $\vec{\alpha}$. The variables x_n also form a vector \vec{x} .

An alternative approach which can be used is to express the operator as a matrix which then multiplies a vector tuple of inputs and whose product yields the output vector. In refs. [6, 3, 7] the notation \hat{O}_f is used for operators. Here we shall keep to a representational notation, such that, if we think of a function f then this has an operator representation $\llbracket f \rrbracket$; thus an operator expression from these references of the form

$$\vec{\alpha} = \hat{e} \vec{x}. \quad (6)$$

would be written

$$\langle \alpha_1, .. \rangle = \llbracket e \rrbracket (\langle x_1, .. x_n \rangle) \quad (7)$$

here. A potential limitation of the matrix form used below and in refs. [6, 3, 7] is that we must have $\dim(\vec{\alpha}) = \dim(\vec{x})$, but this is sufficient to discuss appropriate fragments of the configuration.

2.4.1 Web Server Configuration

An example of a configuration $Q \in \mathcal{C}$, which has no natural expression at the level of bits is the configuration of a web service, using a piece of software, e.g. an HTTP web sever.

A web server is a piece of software whose behaviour is partly programmed, which partly depends on the operating system (or virtual machine) on which it runs, and which admits a number of alternative behaviours that are configurable in it ‘configuration file’, which is an ASCII file called something like `httpd.conf`. This file contains directives such as the following examples:

```
Port 80
HostnameLookups on
LoadModule access_module mod_access.so
LoadModule auth_module mod_auth.so
```

These lines tell the software which communications port to use, whether or not it should try to look up clients in a host-name database and which additional behaviour-modifying modules to load and use.

A policy might constrain this configuration by asking two promises from the file:

- A web server promise to use port 80
- A web server promise that `HostnameLookups` will be “off”.

In the latter case, the excerpt above would have to be edited in order to change value of the second line. For this we need some specialized operator that brings the system back into line with the promise compliant state p

$$[[\text{HostnameLookups}(\text{off})]] \vec{q} = \vec{p} \quad (8)$$

by editing the file. This operator should then be able to search through the file’s internal representation for the line concerned, read it, check its value and, if necessary, rewrite it so that it complies with the policy constraint. The assumption then is that this so-configured text will cause the web server to behave as advertised, i.e. stop trying to resolve the names of hosts.

This example is just one of many similar cases in use today. The current trend in modern computing systems is to use context free languages, such as XML, to represent configuration information in a more structure form. The principal of is the same, but the language belongs to a different place in the Chomsky hierarchy, that of context free languages. This leads to additional computation.

Before leaving this example, notice a matter of some importance about causality. The configuration decisions cached and catalogued in this file apply only to the web server process that reads and uses it, but the resulting behaviour of the web server can affect many other parallel processes in the system – and not just those that use its services. The use of port 80 might block its use by another program. The choice to look up host names in a directory could place a burden on load on the directory service. In other words, it is far from clear that the main hypothesis (that correct policy leads to correct configuration leads to correct behaviour) is true if one follows all of the causal threads of this case.

2.4.2 Example: A trivial fragment in \mathbb{B}^3

Consider now the particular example of Unix file permissions. Under Unix, file access rights have a rather simple representation in terms of bit strings. A user’s file permissions are stored in three bits ‘r’, ‘w’ and ‘x’, standing for read, write and execute access rights respectively. (Similar bits are used for groups and for other users, but it is sufficient to consider the user or owner of a file here.) Here are some examples:

```

rwx
100 Read only
110 Read-write
010 Write only
001 Execute only
```

If one of these bits is set to ‘1’, permission to perform the appropriate action is granted, otherwise it is denied. The operator which changes these permissions is contained within the Unix command ‘chmod’: the command ‘+r’, for instance,

would set the ‘r’ bit, whereas the command ‘-r’ would unset it. We need a notation to describe this. Here we shall use the notation $\llbracket X \rrbracket$ to denote a representation of the operator X :

$$\llbracket +\mathbf{r} \rrbracket \Leftrightarrow \text{“chmod} + \mathbf{r} \text{”} \quad (9)$$

These operations can be represented as *operators*, but how shall we represent them?

Consider first the bit strings that represent the configuration of a file object. Here are two different ways of representing these bit strings: a *vector* representation

$$\begin{pmatrix} 1 \\ r \\ w \\ x \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 0 \end{pmatrix} \quad (10)$$

and a representation as a tuple of *Boolean expressions*.

$$\langle r, w, x \rangle = \langle 1, 1, 0 \rangle. \quad (11)$$

In the vector case, there is an extra ‘1’, whose role becomes apparent during the matrix multiplication below. What do the operators look like to, say, open a file object for reading?

Regardless of representation, such an operator must set the ‘read bit’, while leaving the others unchanged. For the first representation, we can construct this by matrix multiplication. We write this in all its notations:

$$\vec{p} = \llbracket +\mathbf{r} \rrbracket(\vec{q}) = \llbracket \text{chmod} + \mathbf{r} \rrbracket(\vec{q}) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ r \\ w \\ x \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ w \\ x \end{pmatrix} \quad (12)$$

We start with generic values $r, w, x \in \{1, 0\}$ and end up with policy conformant values. The ‘r’ bit is set to 1, regardless of its previous value, and all other bits are kept the same.

2.4.3 File Permissions

For the second representation, we can use a logical OR operation:

$$\begin{aligned} \langle p_r, p_w, p_x \rangle &= \text{chmod} + \mathbf{r}(\langle r, w, x \rangle) \\ &= \llbracket +\mathbf{r} \rrbracket \langle r, w, x \rangle \\ &= \langle r, w, x \rangle \text{ OR } \langle 1, 0, 0 \rangle \\ &= \langle 1, w, x \rangle. \end{aligned} \quad (13)$$

Representation of data lies at the heart of modern computing. Digital computers use bits at the bottom-most layer and programmers build layers of data types on top of these in order to organise and normalise information. Thus, our problem reduces to the manipulation of strings of bits by operators. We shall refer to this as alphabetic configuration management, since each bit string of length n represents, in essence, the assignment of a symbol in a 2^n long alphabet Σ .

2.5 Implementability and Maintainability

At we mentioned in section 1.2, we can attempt to implement policy in two ways: by either starting from a known state and specifying the specific sequence of changes

to get to the result, or we can specify the final state and look for a set of operators which are not hindered from getting to that state. The former method is fragile to single obstacles in a chain of pre-conditions from the initial state to the final state; one must deal with the problem of operator composition.

We shall not consider this approach, as it is discussed in the chapter by Sun and Couch. Rather we shall define configuration in terms of goals, or post-conditions. The idea is then that one should quickly arrive at the policy-compliant state and remain there. If the operator is applied to an already compliant system, it must not lead to any change. Thus the key to describing configuration management in these terms is to look for special properties of the operators.

There is actually a deeper point to this. We claim that maintenance is something that must be repeated because we cannot entirely predict all of the changes that will occur in a configuration fragment. But if we are repeating operations with unclear knowledge of the initial state, how do we know that those operations are not actually harming the system?

Suppose during the maintenance of the Sistine Chapel, a painter is asked to add a fresh coat of paint to Michelangelo's elaborate ceiling decorations. If the task is done only once, it would last a few years but the ceiling would soon need painting again. However, if the artist painted continually, layer upon layer, the paint would become so thick as to fill the room. This operation might even interfere with other operations, such as cleaning the floor. This is not the desired result.

A model of interest is the so-called 'immunity model'[11, 3] of maintenance. The approach used in the immunity model is to break policy into a number of tasks or issues that are *primitive* in the following sense:

- A number of operational *types* is defined, that identify high level system properties. These are coded with configuration symbols that do not overlap.
- An operation of a given type now leads to a single change of state in the bit string that represents that type.

Operations hence form a collection agents for independent changes, with distinct types, which can be applied in an order independent fashion (since the states themselves do not overlap the operators must commute).

Thus, regardless of the logistics of implementation, this post-conditional approach to configuration requires a spanning set of *operators*, which act on alphabetic patterns of configuration data at the appropriate level of coding. To capture the idea that we should end up with a predictable result, there are two properties of interest that we can give to these operators:

- *Idempotence*: the repetition of an operation leads to no change after the initial application. i.e. $f(f(x)) = f(x)$, or for any state \vec{q} , $\llbracket O \rrbracket^2 \vec{q} = \llbracket O \rrbracket \vec{q}$. This contains the essence of non-repetition, but it is a property without an anchor – it does not lead to a predictable post-condition, unless we also know the initial state. So this property has the special status of a bridge between the two approaches to configuration management.
- *Convergence at a fixed point*: the first application of an operation leads to the specified policy result, regardless of the initial state. Thereafter, no further changes should take place. i.e. for any initial state \vec{q} and a policy-compliant state \vec{p} , $\llbracket O \rrbracket^2 \vec{q} = \llbracket O \rrbracket \vec{q} = \vec{p}$. This is the approach preferred by cfengine[11, 25].

Other properties might also be of interest to us. For instance, *injectivity* of an operator tells us whether it has a unique inverse (is it a one-to-many map). This is

a property that some system administrators value, since it means that operations can be ‘rolled back’ to their former state, after a change.

The approach we pursue below is to search for operators that are themselves equipped with the properties above. We shall

2.6 The algorithmic complexity of configuration management

Thus we arrive at the key question we wish to address in this chapter: what is the computational *complexity* of configuration management and what do we mean by this?

As we have seen, one can easily take individual examples of configuration *operations*, as we did in the previous section, and say that configuration management is easy. The algorithms required to make changes are clearly simple things. If they weren’t, computers would not exist as we know them. Implementation is simply a matter of writing data into data structures, at some appropriate level of coding, e.g. changing the permissions of a file, editing its contents, starting and stopping programs etc. Implementing a constant, predecided policy is therefore straightforward. But how do we determine whether a policy exists that leads to the behaviour we require?

If we formulate this question in terms of pre-conditioned sequences, then we must deal with a lot of combinatoric complexity. See the chapter by Sun and Couch for a discussion of this problem. If, on the other hand, we use an approach based on operators that are assumed to be orthogonal and fixed-point convergent, the question is then: can we identify operators that verifiably have the desired policy behaviour in a reasonable length of time?

If we have to determine the implementation method in advance, as part of our estimation of complexity, then the complexity of the task grows considerably from the application of a single transformation to a search in a potentially huge space of possible operations. The result we shall find is the result of a somewhat dumb and blind, brute-force search through this space, by a systematic and somewhat dogged agent. Clearly, one could narrow the search and reduce the complexity if one could inject some intelligence in the form of additional constraints. But that is not what we want to do. We want to know how difficult the whole problem is, from the beginning.

The question we are asking then is not how long does the configuration management software take to run: rather it is, do we know how we write the configuration management software in the first place? The difficulty with system administration lies in determining the correct configuration, satisfying the constraints of policy in an efficient manner.

In the remainder of this chapter, we wish to show that even simple alphabetic configuration problems are, in general at least NP hard.

3 A gentle introduction to complexity theory

Let us take a step backward in order to move forward and study some very simple problems to understand the meaning of complexity.

3.1 What is a ‘problem’?

In complexity theory a *problem* is a question to which the answer is YES or NO. The question has an *input*, and *solving* the problem amounts to providing the correct answer to the question when a particular input is given. Let us study some examples.

(Problem **SEARCH**) *Input:* a natural number n and a list of numbers ℓ_1, \dots, ℓ_k . *Question:* Does the number n occur in the list ℓ_1, \dots, ℓ_k ?

The input should be expressed in some fixed alphabet according to some sensible convention. For the problem *A* we may e.g. use the alphabet

0 , 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , ! , /

and the convention that the input should be given on the form $n!\ell_1/\dots/\ell_k$ where the natural numbers n, ℓ_1, \dots, ℓ_k are given by decimal digits without leading zeros. Thus, e.g. 27!17/0/23961/27/61 and 31!71/9/961 are admissible; for the former input, the answer to the question is YES, for the latter, the answer is NO.

The input to a problem should always be a string of symbols over some fixed alphabet, and the representation of the input should always follow some fixed convention. However, as long as we make natural and reasonable choices, it does not matter which particular alphabet, or which particular convention, we choose. These choices have no essential mathematical implications. Thus, when a problem is stated in complexity theory, the exact alphabet and convention are often suppressed. The reader is free to choose his³ favourite representation in his favourite alphabet. (He has to be reasonable, though. A typical unreasonable representation of the input of problem *A*, will be that the m 'th number in the list should be given with 2^m leading zeros.)

Intuitively, the problem **SEARCH** is very easy to solve. We just have to traverse the ℓ_1, \dots, ℓ_k ; if the number we are looking for, that is n , is there, the answer is YES; if the number is not there, the answer is NO. Let us look at an apparently harder problem.

(Problem **SPLIT**) *Input:* a set of natural numbers X . *Question:* Is it possible to split X into two sets Y and Z such that the sum of the numbers in Y equals the sum of the numbers in Z .

If the input the problem is the set $\{1, 2, 3, 4, 5, 7\}$, the answer to the question is YES because $1 + 2 + 3 + 5 = 4 + 7$. Now, what is the answer when the input is

$\{3, 6, 9, 10, 20, 30, 173, 174, 175, 1027, 1058, 1132, 1719, 2480\}$?

No doubt, intuitively problem **SPLIT** seems harder to solve than problem **SEARCH**.

3.2 The class P

Definition 2 (The class P) Let f be any function from \mathbb{N} to \mathbb{N} and let $|x|$ denote the number of alphabet symbols required to represent the input x . An algorithm taking x as input works in time $f(|x|)$ if $f(|x|)$ bounds the number of basic steps executed by the algorithm. We will say that an algorithm works in polynomial time if there exists a polynomial p such that the algorithm works in time $p(|x|)$. A problem belongs to the class P if (and only if) the problem can be solved by an algorithm working in polynomial time. We use standard terminology when say that the problems in P are the problems solvable, or decidable, in polynomial time.

³We assume that women are too complex to be represented here.

The class of problems solvable in polynomial time, called P , is well known from the literature, and the class will be introduced at an early stage in any basic course in complexity theory. The problems in the class are seen as the tractable problems, that is, if a problem belongs to P , a computer can solve the problem in reasonable time; if a problem does not belong to P , the problem will be intractable in the sense that it require more time to solve the problem than any practical application can afford.

At this stage, the practically-minded reader should ask how reasonable it is to identify the tractable problems and the polynomial time decidable problems, and the mathematically-minded reader should find the definition of P somewhat suspicious. (Hopefully, the two groups of readers are not disjoint.)

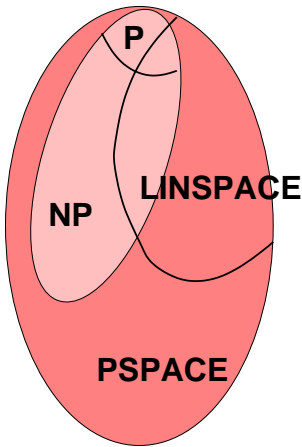


Figure 4: The complexity classes in the space of algorithms.

Unfortunately for the the practically-minded reader, it is beyond the scope of this paper to discuss his question any further, but such a discussion can be found in numerous text books and survey papers, e.g. see [2] and [1]. These discussions will assure the reader that the formally defined class P capture the informal class of tractable problems surprisingly well; if we can prove that a natural problem belongs to P , we can be pretty sure that we can implement an efficient algorithm solving the problem; if we can prove that a problem does not belong to P , no real-life computer can deal with the problem no matter how powerful it might be. Of course, no one asserts that algorithm executing, let us say, $|x|^{(2^{100})}$ basic steps terminates in reasonable time, still, such an algorithm runs in polynomial time. However, it turns that most *natural* problem that can be solved in polynomial time, actually can be solved in time $p(|x|)$ where p is a polynomial of low degree, typically 2 or 3.

The mathematically-minded reader should worry since the definition of the class P is a bit sloppy. The definition does not say what a *basic step in the execution of an algorithm* is. Neither does the definitions say what an *algorithm* is. Well, in order to help the reader to grasp the essence of polynomial time computability, we have chosen to rely on slightly informal notions that, strictly speaking, should not appear in a rigorous definition. We do so because any computer scientist or programmer knows what algorithm is. It is an efficient and unambiguous receipt for transforming input into output. It is something you can implement on your desktop computer if the computer is equipped with sufficiently memory. The reader probably has a very strong sense of what algorithm is even if he cannot come up with the exact formal definition, like e.g. most educated people know what a real number is even if they are not familiar the technical definitions occurring in the mathematical literature.

In complexity theory, algorithms are usually defined by Turing machines. A Turing machine is a formally defined mathematical entity that can be viewed as a mechanical device manipulating symbols on an infinite tape. The informal notion of an algorithm are captured by Turing machines in the following sense: (i) *any procedure we intuitively conceive as an algorithm, can be implemented on a Turing machine* and (ii) *any computation carried out by a Turing machine is an algorithm in the intuitive sense*. It is straightforward to define the basic steps of a Turing machine since you can point to something, declare that *this is a basic step* and this is it, e.g. to write a symbol on the tape counts as one step, to move the machine's scanning head one position counts as one step, etc. It is not that easy to identify the basic steps when we are dealing with informal algorithms, but in every algorithm there will be some operations that are natural candidates, e.g. if the algorithm is given as a Java program, the basic instructions you find in the Java manual are the natural candidates; if the algorithm is a clerical routine meant to be carried out by pen and paper, to write down a symbol is a natural candidate. P is a robust class of problems, and as long as our definition of a step in an algorithm is reasonable, the class will not be sensitive to the details of the definition. For sure, it might very well be the case that a particular problem is solvable in time, let us say, $(x + 1)^2 + 17$ if one step amounts to carry out one instruction in a high level programming language, but not in time $(x + 1)^2 + 17$ if one step amounts to carry out one instruction on a rudimentary device like a Turing machine, still, the problem would be solvable in *polynomial time* by any reasonable rudimentary device you can imagine, maybe in time $(x + 2)^3$, maybe in time $x^{23} + 1000$.

Thus, we have chosen to skip the detour via Turing machines, and thus some dull technical definition, and rely on the intuitive notion of an algorithm in our definition of the class P.

3.3 Proof techniques

Before trying to write a program solving a particular problem, a programmer should ask: is the problem in P? If the answer is no, the problem is essentially intractable for any computer no matter how powerful, and the programmer's task is in some sense insurmountable. (It is beyond the scope of this paper what a programmer should do in such a situation, but he will have to resort to various advanced techniques and methods for finding approximate solutions.) Now, how can a programmer establish if the problem he encounters is in P or not?

To prove that a problem is in P, you simply have to find an algorithm solving the problem in polynomial time. It might be hard to find the algorithm, it might be hard to prove that the algorithm indeed solves the problem, and it might be hard to prove that the algorithm indeed works in polynomial time, but it is easy to understand the overall structure of such a proof. The definition of P says that the problem is in P if and only if there exists an algorithm solving the problem in polynomial time. If you can provide such an algorithm, you have proved that the problem is in P.

In the remainder of this section we will discuss how we can prove that a problem does not belong to P. We will demonstrate the standard proof technique.

Imagine you cannot come up with a polynomial time algorithm solving a particular problem X in polynomial time. Can you conclude that X does not belong to P? Of course not. Even if you are an experienced and clever programmer, you cannot conclude that such an algorithm does not exist just because you cannot come up with one. How should you argue, then? There are infinitely many algorithms, and you cannot go through each and one of them and prove that this doesn't work, and

this doesn't work, and *this* doesn't work, and so on; on your dying day there will still be infinitely many algorithms left to consider. (Besides, imagine what a boring life.) Now, what you need is a problem that you already know does not long to P. Let us denote this problem by H for HARD, and you may e.g. find H in a textbook where you also find a theorem stating that

$$\text{no algorithm can solve } H \text{ in polynomial time.} \quad (\dagger)$$

Then you can try to prove that

$$\begin{array}{l} \text{if there is an algorithm solving } X \text{ in polynomial time,} \\ \text{then there is an algorithm solving } H \text{ in polynomial time.} \end{array} \quad (\ddagger)$$

If you succeed, you can conclude beyond any doubt and with mathematical certainty that there is no algorithm solving X in polynomial time, that is, X does not belong to P. Your argument has the structure

$$\frac{\text{If } p \text{ then } q. \quad \text{Not } q.}{\text{Not } p.} \quad \begin{array}{l} \text{(premises)} \\ \text{(conclusion)} \end{array}$$

Numerous mathematical proofs embody this structure. The obviously sound inference is of course frequently used in daily life reasoning too, and it has even got its own name as the medieval philosophers baptised the inference *modus tollens*.

So far, so good. If you can prove (\ddagger) , then you are done, but how should you proceed to prove (\ddagger) . This is where the pivotal notion of polynomial time reducibility enters the stage.

Definition 3 (Polynomial time reducibility) A function f polynomial time reduces a problem A to a problem B if (and only if)

1. the answer to question A with input x is the same as the answer to question B with input $f(x)$
2. the function f is computable in polynomial time.

If there exists a function f which polynomial-time reduces the problem A to the problem B , we will say that A is polynomial time reducible to B .

If you can find a function f such that f polynomial time reduces H to X , then (\ddagger) follows. Given such a reduction function, you can solve problem H on input x by first computing the value $v = f(x)$ and thereafter solve the problem X on input v . (The answer to problem X on input v will be the same as the answer to problem H on input x .) The number of basic steps needed to compute $f(x)$ is bounded by a polynomial in the length of the input, let us say $p(|x|)$. The length of the output, that is v , will also be bounded by a polynomial $p'(|x|)$ since the length of the output of any computation taking place in polynomial time will be bounded by a polynomial in the input. If the number of basic steps required to solve X also is bounded by a polynomial in the length of the input, let us say $q(|v|)$, then the whole computation will take place in polynomial time. We have

$$\overbrace{p(|x|)}^{\text{solve } H \text{ on input } x} + \underbrace{q(|v|)}_{\substack{\text{compute} \\ v = f(x)}} \leq p(|x|) + q(p'(|x|))$$

and the expression $p(|x|) + q(p'(|x|))$ is a polynomial in the length of the input. Thus, if there is a algorithm solving X in polynomial time, there will also be one solving H in polynomial time, i.e. (\ddagger) holds, and when (\ddagger) holds, it follows that X does not belong to P .

We conclude the section by a brief summary: To prove that a problem X belongs to P , you should simply provide an algorithm solving X in polynomial time. To prove that a problem X does not belong to P , you should (i) find a problem H that does not belong to P and (ii) prove that H is polynomial time reducible to X .

3.4 A hard problem

Let us formulate one of the main insights of the previous section as a theorem.

Theorem 1 *Let A and B be problems such that (i) A does not belong to P , and (ii) A is polynomial time reducible to B . Then, B does not belong to P .*

In Theorem 1 we find the contours of an essential property of any sensible measure: *If A is far away, and B is at least as far away as A , then B is also far away.* We are measuring complexity. If a problem A is polynomial time reducible to a problem B , then it will be as least as hard to solve B as it will be to solve A (modulo polynomial time). Thus, if A is far away from the tractable, so is also B .

We will apply Theorem 1 to argue that several natural problems on operators are intractable. Hence, the B in the theorem will be instantiated by different problems on operators whereas the A might e.g. be instantiated by a celebrated hard problem on boolean algebra, namely the problem **SAT**.

(Problem **SAT**) *Input:* a boolean expression α . *Question:* Is α satisfiable?

Presumably most of the readers will to some extent be familiar with boolean algebra, still, let us re-examine the subject briefly to assure that everyone understands the the problem **SAT**. First we define the boolean expressions.

Definition 4 (Boolean expressions) *We have an infinite supply of boolean variables x_0, x_1, x_2, \dots*

- *A boolean variable is a boolean expression.*
- *The constants 0 and 1 are boolean expressions.*
- *$(\alpha \cdot \beta)$ is a boolean expression if α and β are boolean expressions.*
- *$(\alpha + \beta)$ is a boolean expression if α and β are boolean expressions.*
- *$\bar{\alpha}$ is a boolean expression if α is a boolean expression.*

Further, we will use standard terminology and say that $(\alpha \cdot \beta)$ is the product of α and β ; that $(\alpha + \beta)$ is the sum of α and β ; that $\bar{\alpha}$ is the complement of α .

As we say that $(\alpha \cdot \beta)$ and $(\alpha + \beta)$ are respectively a *product* and a *sum*, it would also be natural to say \cdot and $+$ are respectively the *multiplication* operator and the *addition operator*. This is standard terminology, but the reader should be aware that the two operators also frequently are referred to as the **AND**-operator and the

OR-operator in the literature. The reader might even be familiar with propositional logic where we have two truth values **false** and **true** and three connectives AND, OR and NOT. Boolean algebra is nothing but propositional logic, and

- 0 and 1 correspond to respectively **false** and **true**
- \cdot and $+$ correspond to respectively AND and OR
- $\bar{\alpha}$ corresponds to NOT α .

Definition 4 gives the form of an boolean expression. According to the definition e.g.

$$(x_3 + \overline{(x_{17} \cdot x_2)}) \quad \text{and} \quad \overline{((x_9 + \overline{(1 + x_2)}) \cdot (x_4 \cdot \bar{x}_4))} \quad (14)$$

are boolean expression. Now, a boolean variable either takes the value 0 or the value 1, and given an assignment of $\{0, 1\}$ values to the variable, we can compute the value of a boolean expression. To do so, we proceed as we do when we compute the value of an arithmetical expression, but we stick to the following rules:

$$\begin{array}{llll} 0 + 0 & = & 0 & 0 \times 0 & = & 0 \\ 0 + 1 & = & 1 & 0 \times 1 & = & 0 & \bar{0} & = & 1 \\ 1 + 0 & = & 1 & 1 \times 0 & = & 0 & \bar{1} & = & 0 \\ 1 + 1 & = & 1 & 1 \times 1 & = & 1 \end{array}$$

Hence, since the sum and the product work as in ordinary arithmetic except that $1 + 1$ equals 1, the value of a boolean expression will always turn out to be 0 or 1, e.g. given the assignment

$$x_3 := 0, \quad x_{17} := 1, \quad x_2 := 1$$

we have $(x_3 + \overline{(x_{17} \cdot x_2)}) = 0$; whereas given the assignment

$$x_3 := 1, \quad x_{17} := 0, \quad x_2 := 1 \quad (15)$$

we have $(x_3 + \overline{(x_{17} \cdot x_2)}) = 1$.

Definition 5 (Satisfiability) *A boolean expression α is satisfiable if there exists an assignment to the variables in α such that $\alpha = 1$.*

Hence, the expression $(x_3 + \overline{(x_{17} \cdot x_2)})$ is satisfiable since the expression equals 1 under the assignment (15). The rightmost expression in (14) is not satisfiable.

This should settle what the problem **SAT** is all about, but why should the problem be hard, on the surface it might look like an innocent little problem? To get some intuition, the reader should observe that the number of possible assignments is exponential in the number of variables in the expression. If there are n variables in an expression, there will be 2^n different assignments, and to solve the problem **SAT** we might be forced to consider each and one them. We will simply not be able to do so when n is large, if $n = 100$, we will not be done in a million years even if we checked a million assignments per second.

But why should we be forced to checked each and one of the 2^n assignments? Well, there are algorithms solving **SAT** that in certain respects are far more efficient than the straightforward stupid brute force search for an satisfying assignment, however, to make a long story short, none has ever been able to come up with an algorithm solving the problem in polynomial time, and if you can come up with one, there is

a million dollar award waiting for you. The problem is known to be NP-complete, that is, if **SAT** is solvable in polynomial time, every problem in the complexity class called NP will be solvable in polynomial time, and that is considered as highly unlikely among all sane complexity-theorists, and probably among a few insane ones too. Though we lack a mathematical proof, there are absolutely no evidence there exists an algorithm solving **SAT** in polynomial time.

We will treat the class NP more thoroughly in Section 5, still, let us mention right now that the standard definition says that NP is the class of problems solvable by a *non-deterministic Turing machine* working in polynomial time. For reasons explained above, a slightly sloppy, but rather informative alternative definition would be the class of problems solvable by a *non-deterministic algorithm* working in polynomial time (but a deterministic algorithm would have to use exponential time). The hardest problems in NP are the NP-complete ones. In addition to **SAT**, hundreds of problems from science, mathematics and daily life are known to be NP-complete, among them our example **SPLIT** discussed in Section 3.1. If you either can prove that $P = NP$ or that $P \neq NP$, the million dollar award mentioned above is yours. Those of you in urgent need of money, should take our advice and not waste your time on the former option. Start working right away on the latter.

4 Analysis of some configuration problems

4.1 The complexity of the system coding

We asked the question earlier: what is the effect of the high level coding on the complexity of configuration management? This is an important question because we want to argue that a model of bit strings is adequate to discuss the complexity of the whole problem. We can now make a plausible answer to this question.

The complexity of coding has two facets: (i) the complexity of the syntactic parsing and translation, (ii) the semantic translation - how do we find identify the behaviour associated with the strings in the high level coding?

For (i), the following result is fundamental in parsing theory[26, 27].

Theorem 2 *Given a context free grammar G and a string s , deciding whether the string s is an element of the language \mathcal{L}_G generated by G can be done in $O(|G| \times |s|^3)$ deterministic time and $O(|G| \times s^2)$ space.*

This complexity does not increase even for extended context free grammars. In other words, within polynomial time, we can translate the syntax of any reasonable high level coding, so this will not affect the answer as to whether configuration management is hard or not.

For (ii), the second of these points is not a question we can answer with any certainty, but we believe that whatever the answer is, any contribution to behaviour as a result of the coding belongs in the definition of the problem itself and is therefore a valid part of the answer we seek. With this assumption, we can proceed to view configuration management as a problem of determining operators represented in \mathbb{B}^n .

There is a large literature on coding, and its algorithmic complexity. See ref. [28, 24] for a sample. The usual problem addressed in coding is that posed by Shannon for information transmitted over a noisy channel, namely given a transmitted code word and a noisy or uncertain channel, what is the complexity of determining the original

transmitted word, e.g. by maximum probability? This problem is known to be NP complete. The configuration problem is essentially related to this scenario[19] in two stages: (i) given a result in the configuration alphabet observed at the end of the policy-configuration channel, what policy does this correspond to? and (ii) given an observed behaviour at the end of the configuration-behaviour channel, what is the maximum likelihood configuration that led to this?

4.2 Operators formalised

Our goal now is to undertake a complexity-theoretic analysis of some problems occurring naturally in the theory of system administration and configuration management, using the model of operators over bit strings. A complexity-theoretic analysis requires that we formalise the operators such that we know exactly what we are talking about and can state the problems with sufficiently of mathematical precision. In this section will carry out such a formalisation. How well our formalisation captures the operators as they appear in the theory of system administration, will not be discussed any further, but based on the discussion above, we believe our formalisation to be reasonable.

Recall that \mathbb{B}^n is the set of all bit strings of length n , e.g.

$$\mathbb{B}^4 = \{0000, 0001, 0010, 0011, 0100, 0101, 0110, 0111, 1000, 1001, 1010, 1011, 1100, 1101, 1110, 1111\}.$$

If $b \in \mathbb{B}^n$, then b_i denotes the i 'th bit in b . We will count the bits in a string from the left to the right, thus if $b = 0111$, then $b_1 = 0$, $b_2 = 1$, $b_3 = 1$ and $b_4 = 1$.

An operator is nothing but a total function with domain \mathbb{B}^n and range \mathbb{B}^n . So, an operator transforms a string of bits into another string of bits of the same length. Let us study a few examples. The operator INV inverts the bits of a string, thus we have e.g. $\text{INV}(0111) = 1000$ and $\text{INV}(1001) = 0110$. The operator RST sets all the bit in a string to 0, and we have e.g. $\text{RST}(0111) = 0000$ and $\text{RST}(1001) = 0000$. The operator SUC increases the binary number given by a string by 1 modulo 16. We have $\text{SUC}(0000) = 0001$, $\text{SUC}(0001) = 0010$, $\text{SUC}(0010) = 0011$, \dots , $\text{SUC}(1111) = 0000$.

Even if there are only finitely many operators over the domain \mathbb{B}^n , there are still a lot of them. The exact number of operator over the domain \mathbb{B}^n is 2^{n2^n} . There are more that 10^{19} operators over the domain \mathbb{B}^4 , and there are more operators over the domain \mathbb{B}^8 than there are elementary particles in the universe. We need a language strong enough to express all the operators over the domain \mathbb{B}^n in a convenient and uniform way. That is our motivation for introducing the operator expressions.

Why are there 2^{n2^n} operators over the domain \mathbb{B}^n ? Let A and B be a finite sets containing respectively m and n elements. The number of functions with domain A and range B will be n^m . The set \mathbb{B}^n contains 2^n elements, and an operator over the domain \mathbb{B}^n is a function with domain \mathbb{B}^n and range \mathbb{B}^n . Thus, by applying the number-theoretic equality $(a^b)^c = a^{bc}$, we see that the number of operators over the domain \mathbb{B}^n will be

$$(2^n)^{(2^n)} = 2^{n2^n}.$$

Another way of saying this is that 2^n is the size σ of an effective alphabet of configurations Σ , that can be represented in n bits, so 2^{n2^n} is σ^σ the number of "commands" or algorithms one can form to transform one string in this alphabet to another (see fig 5).

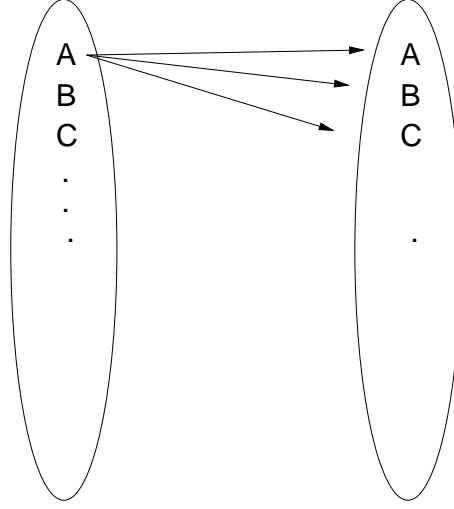


Figure 5: The collection of all possible mappings of σ configurations is σ^σ large.

Definition 6 (Operator Expressions) *An operator expression e is a sequence $\langle \alpha_1, \alpha_2, \dots, \alpha_n \rangle$ of boolean expressions. If there are n boolean expression in the sequence, the only variables allowed in the expressions are x_1, x_2, \dots, x_n .*

An operator expression $e \equiv \langle \alpha_1, \alpha_2, \dots, \alpha_n \rangle$ defines an operator over the domain \mathbb{B}^n . As explained, we use $\llbracket e \rrbracket$ to denote the operator defined by the operator expression e . The value $\llbracket e \rrbracket(b)$, that is, the value of $\llbracket e \rrbracket(b)$ in the particular argument $b \in \mathbb{B}^n$ is given by the value of the sequence $\langle \alpha_1, \dots, \alpha_n \rangle$ under the assignment $x_1 := b_1, \dots, x_n := b_n$. The first bit of $\llbracket e \rrbracket(b)$ is given by the value of α_1 under the assignment, the second bit by the value of α_2 , and so on. We better look at a few examples.

For instance, let $e \equiv \langle \overline{x_1}, \overline{x_2}, \overline{x_3}, \overline{x_4} \rangle$, then $\llbracket e \rrbracket$ will be the operator INV inverting a bit string of length 4. Further, we can compute the value $\llbracket e \rrbracket(0011)$ by computing each of the boolean expressions in the sequence $\langle \overline{0}, \overline{0}, \overline{1}, \overline{1} \rangle$.

A more sophisticated example will be if

$$e \equiv \langle ((\overline{x_1} \cdot ((x_2 \cdot x_3) \cdot x_4)) + (x_1 \cdot \overline{((x_2 \cdot x_3) \cdot x_4)})), (\overline{x_2} \cdot (x_3 \cdot x_4)) + (x_2 \cdot \overline{(x_3 \cdot x_4)}), (\overline{x_3} \cdot x_4) + (x_3 \cdot \overline{x_4}), \overline{x_4} \rangle.$$

Now, let us compute the value $\llbracket e \rrbracket(0011)$. First we make the assignment

$$x_1 := 0, x_2 := 0, x_3 := 1, x_4 := 1$$

and then we evaluate the boolean expressions in the sequence one by one. We have

$$((\overline{x_1} \cdot ((x_2 \cdot x_3) \cdot x_4)) + (x_1 \cdot \overline{((x_2 \cdot x_3) \cdot x_4)})) = ((\overline{0} \cdot ((0 \cdot 1) \cdot 1)) + (0 \cdot \overline{((0 \cdot 1) \cdot 1)})) = 0$$

and thus the first bit in the value is 0; we have

$$(\overline{x_2} \cdot (x_3 \cdot x_4)) + (x_2 \cdot \overline{(x_3 \cdot x_4)}) = (\overline{0} \cdot (1 \cdot 1)) + (0 \cdot \overline{(1 \cdot 1)}) = 1$$

and thus the second bit in the value is 1; we have

$$(\overline{x_3} \cdot x_4) + (x_3 \cdot \overline{x_4}) = (\overline{1} \cdot 1) + (1 \cdot \overline{1}) = 0$$

and the third bit turns out to be 0; the forth bit also turns out to be 0 since $\bar{x}_4 = \bar{1} = 0$. Hence, $\llbracket e \rrbracket(0011) = 0100$. It might be an instructive exercise for the reader to verify that $\llbracket e \rrbracket$ indeed is the operator SUC discussed above, and that $\llbracket e \rrbracket(0000) = 0001$, $\llbracket e \rrbracket(0001) = 0010$, and so on. In such a situation we might use the notation $\llbracket e \rrbracket = \text{SUC}$ and say that the expression e *defines* the operator SUC. Observe that many expression will define the same operator, e.g. the operator RST can be defined by the expression $\langle 0, 0, 0, 0 \rangle$, but also by the expression $\langle (x_1 \cdot \bar{x}_1), (x_1 \cdot \bar{x}_1), (x_1 \cdot \bar{x}_1), (x_1 \cdot \bar{x}_1) \rangle$ and the expression $\langle \bar{1}, (x_1 \cdot \bar{x}_1), (x_4 \cdot \bar{x}_4), 0 \rangle$, indeed, there will be infinitely many expressions defining any operator.

4.3 Some problems on operators: IDM, INJ and CON

Having formalised the notion of an operator, we are now ready to state some problem (see fig. 6).

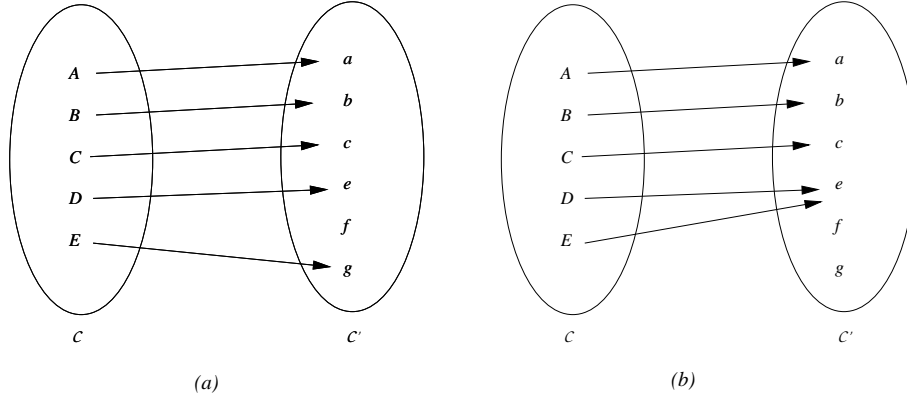


Figure 6: Operator mapping properties. (a) An injective map has a unique inverse; (b) A non-injective map does not have a unique inverse. In both cases a operator transformation could eliminate parts of the configuration space from the domain of possibility, i.e. both represent potential constraints.

(Problem **IDM**) *Input:* an operator expression e . *Question:* Is $\llbracket e \rrbracket$ idempotent, i.e. do we have $\llbracket e \rrbracket(\llbracket e \rrbracket(x)) = \llbracket e \rrbracket(x)$ for all $x \in \mathbb{B}^n$?

(Problem **INJ**) *Input:* an operator expression e . *Question:* Is $\llbracket e \rrbracket$ an injection, i.e. do we have $\llbracket e \rrbracket(x) \neq \llbracket e \rrbracket(y)$ if $x \neq y$?

(Problem **CON**) *Input:* an operator expression e . *Question:* IS $\llbracket e \rrbracket$ convergent, i.e. does there, for every $x \in \mathbb{B}^n$, exist $k \in \mathbb{N}$ such that $\llbracket e \rrbracket^k(x) = \llbracket e \rrbracket^{k+1}(x)$?

(Problem **CONx**) *Input:* a pair $\langle e, b \rangle$ where e is an operator expression and $b \in \mathbb{B}^n$. *Question:* Is $\llbracket e \rrbracket$ convergent in b , i.e. does there exist a $k \in \mathbb{N}$ such that $\llbracket e \rrbracket^k(b) = \llbracket e \rrbracket^{k+1}(b)$?

The problem **CON** corresponds roughly to cfengine's notion of convergence at a fixed point, since it claims that (after a certain number of changes have taken place by repeated application of the operator) the result will converge to a specific location

for any input⁴. Hence we are particularly interested in this case for configuration management[3]. What we mean by the complexity in this context is determining whether the policy-operator expression e is convergent (i.e. approaches a fixed and predictable result b) given that it starts in *any given starting state*.

Put another way: we declare the answer b to be policy. This tells us how we would like the system to look after the repeated operation of $\llbracket e \rrbracket$. Then we take the trial method $\llbracket e \rrbracket$, defined by the operator expression e , and we ask: is this policy implementable for every possible start configuration x ? If not, then we have not truly solved the problem with $\llbracket e \rrbracket$.

We will focus on the four problems stated above, but we can easily imagine other interesting problems on operators, e.g.

(Problem *EQUAL*) *Input*: a pair $\langle e, e' \rangle$ where e and e' are operator expressions. *Question*: Does e define the same operator as e' , i.e. do we have $\llbracket e \rrbracket(x) = \llbracket e' \rrbracket(x)$ for all $x \in \mathbb{B}^n$?

(Problem *INVERSE*) *Input*: a pair $\langle e, e' \rangle$ where e and e' are operator expressions. *Question*: Is $\llbracket e' \rrbracket$ the inverse operator of $\llbracket e \rrbracket$, i.e. do we have $\llbracket e \rrbracket(\llbracket e' \rrbracket(x)) = x$ for all $x \in \mathbb{B}^n$?

(Problem *SAME LIMIT*) *Input*: a pair $\langle e, e' \rangle$ where e and e' are operator expressions. *Question*: DO $\llbracket e \rrbracket$ and $\llbracket e' \rrbracket$ converge to the same limit, i.e. will there for any $x \in \mathbb{B}^n$ exist $k, \ell \in \mathbb{N}$ such that

$$\llbracket e \rrbracket^k(x) = \llbracket e \rrbracket^{k+1}(x) = \llbracket e' \rrbracket^\ell(x) = \llbracket e' \rrbracket^{\ell+1}(x) ?$$

(Problem *COMMUTE*) *Input*: a pair $\langle e, e' \rangle$ where e and e' are operator expressions. *Question*: DO $\llbracket e \rrbracket$ and $\llbracket e' \rrbracket$ commute, i.e. do we have $\llbracket e \rrbracket(\llbracket e' \rrbracket(x)) = \llbracket e' \rrbracket(\llbracket e \rrbracket(x))$ for all $x \in \mathbb{B}^n$?

4.4 The Complexity of IDM and INJ

In this section we will show that any algorithm solving the problem **IDM**, or the problem **INJ**, in polynomial time, yields an algorithm solving **SAT** in polynomial time, and if **SAT** does not belong to P, it follows from our results that neither **IDM** nor **INJ** belong to P. Unfortunately, we cannot prove that the problems do not belong to P because we cannot prove that **SAT** does not belong to P, but that is, as we discussed above, highly unlikely, and anyway, even if the world should be so strange that there is an algorithm solving **SAT** in polynomial time, as long as we do not know that algorithm, we are better off living our lives as if **SAT** does not belong to P.

If A denotes a problem, then coA denotes the co-problem. The problem coA is the problem which on any input has the opposite answer of the problem A , that is, the answer to coA on input x is YES if, and only if, the answer to A on input x is NO. Hence, $co\mathbf{IDM}$ is the problem

($co\mathbf{IDM}$) *Input*: an operator expression e .

Question: Do we have $\llbracket e \rrbracket(\llbracket e \rrbracket(b)) \neq \llbracket e \rrbracket(b)$ for some $b \in \mathbb{B}^n$?

⁴It does not specify what the final location is, so the outcome is not as predictable as we would like. Ideally we want $\llbracket e \rrbracket^k(x) = \llbracket e \rrbracket^{k+1}(x) = \text{policy}$ for all x , but from a complexity theoretic viewpoint it is the convergence that is hard to verify, not the point at which the operator converges.

and **coINJ** is the problem

(**coINJ**) *Input:* an operator expression e .

Question: Do we have $\llbracket e \rrbracket(b) = \llbracket e \rrbracket(b')$ and $b \neq b'$ for some $b, b' \in \mathbb{B}^n$?

We will give a very detailed and meticulous proof of the next theorem.

Theorem 3 **SAT** is polynomial time reducible to **coIDM**.

Proof. Definition 3 says that **SAT** is polynomial time reducible to **coIDM** if there exists a function f such that

1. the answer to the question **SAT** on input x is the as the answer to the question **coIDM** on input $f(x)$
2. the function f is computable in polynomial time.

When we can provided such a function f , we have proved the theorem. The input to **SAT** is a boolean expression whereas the input to **coIDM** is an operator expression. Thus, the function f should transform an arbitrary boolean expression α to an operator expression $e = f(\alpha)$ such that

- (1) if α is satisfiable, then $\llbracket e \rrbracket(\llbracket e \rrbracket(b)) \neq \llbracket e \rrbracket(b)$ for some $b \in \mathbb{B}^n$.
- (2) if α is not satisfiable, then $\llbracket e \rrbracket(\llbracket e \rrbracket(b)) = b$ for all $b \in \mathbb{B}^n$.

When we have found such a function f , and argued that f is computable in polynomial time, we are done.

We can, without loss of generality, assume that if a boolean expression contains n different variables, then every variable in the expression occurs in the list x_1, \dots, x_n .⁵ Let INV be the operator inverting the bits of string, and let the operator $\text{OP}_\alpha : \mathbb{B}^n \rightarrow \mathbb{B}^n$ be such that

$$\text{OP}_\alpha(b) = \begin{cases} \text{INV}(b) & \text{if } \alpha = 1 \text{ under either} \\ & \text{the assignment } x_1 := b_1, \dots, x_n := b_n \\ & \text{or the assignment } x_1 := \bar{b}_1, \dots, x_n := \bar{b}_n \\ b & \text{otherwise.} \end{cases}$$

The boolean expression α is a parameter to the operator, and for each α we have an operator OP_α . Let's see how OP_α behaves in (i) the case when α is satisfiable and (ii) the case when α is not satisfiable. In the case (i) when α is satisfiable there will be a least on bit string b such that $\text{OP}_\alpha(b) = \text{INV}(b)$. Observe that if $\text{OP}_\alpha(b) = \text{INV}(b)$, then we also have $\text{OP}_\alpha(\text{INV}(b)) = \text{INV}(\text{INV}(b))$. Further, for any bit sting c we have $\text{INV}(\text{INV}(c)) = c$. Thus, if α is satisfiable, there will be at least one bit sting b such that

$$\text{OP}_\alpha(\text{OP}_\alpha(b)) = \text{INV}(\text{INV}(b)) = b \neq \text{INV}(b) = \text{OP}_\alpha(b)$$

that is, we have $\text{OP}_\alpha(\text{OP}_\alpha(b)) \neq \text{OP}_\alpha(b)$ for some $b \in \mathbb{B}^n$. In the case (ii) when α is not satisfiable we have $\text{OP}_\alpha(b) = b$ for all $b \in \mathbb{B}^n$, and hence also $\text{OP}_\alpha(\text{OP}_\alpha(b)) = \text{OP}_\alpha(b)$ for all $b \in \mathbb{B}^n$.

⁵This is a harmless assumption making life a little bit easier. Let us say that there a three different variables in an expression and that these three variables are x_9, x_{341}, x_5 . By renaming variables in an obvious way we can find an equivalent expression where no variables except x_1, x_2 and x_3 occur.

This shows that if e is an operator expression such that $\llbracket e \rrbracket = \text{OP}_\alpha$, then (1) and (2) will be satisfied. We will proceed as follows. First we will provide a function f transforming a boolean expression α into an operator expression e such that $\llbracket e \rrbracket = \text{OP}_\alpha$. Thereafter we will provide an algorithm computing f and argue that the algorithm runs in polynomial time. That will prove the theorem.

The definition of the reduction function f . Let $\hat{\alpha}$ denote the boolean expression we get when we replace each variable in α by its complement, that is, we replace each occurrence x_i by \bar{x}_i . Further, let

$$\beta_i \equiv ((\bar{x}_i \cdot (\alpha + \hat{\alpha})) + (x_i \cdot \overline{(\alpha + \hat{\alpha})}))$$

for $i = 1, \dots, n$, and finally let $f(\alpha) = \langle \beta_1, \dots, \beta_n \rangle$.

The observation that

$$\beta_i = \begin{cases} \bar{x}_i & \text{if } \alpha = 1 \text{ under either} \\ & \text{the assignment } x_1 := b_1, \dots, x_n := b_n \\ & \text{or the assignment } x_1 := \bar{b}_1, \dots, x_n := \bar{b}_n \\ x_i & \text{otherwise.} \end{cases}$$

makes it easy to see that we indeed have $\llbracket e \rrbracket = \text{OP}_\alpha$ whenever $e = f(\alpha)$. Thus, we can devote the remainder of the proof to argue that f is computable in polynomial time.

There is straightforward algorithm for computing f . The input to the algorithm is of course a boolean expression α . First the algorithm counts the number of different variables in α . Let us say there are n different variables and that the variables occurring in α is x_1, \dots, x_n .⁶ After the counting, the algorithm prints (to the output) the marker symbol \langle , thereafter the algorithm prints

$$((\bar{x}_1 \cdot (\alpha + \hat{\alpha})) + (x_1 \cdot \overline{(\alpha + \hat{\alpha})})),$$

thereafter the algorithm prints

$$((\bar{x}_2 \cdot (\alpha + \hat{\alpha})) + (x_2 \cdot \overline{(\alpha + \hat{\alpha})})),$$

and so on, finally the algorithm prints

$$((\bar{x}_n \cdot (\alpha + \hat{\alpha})) + (x_n \cdot \overline{(\alpha + \hat{\alpha})})) \rangle$$

and terminates. We will argue that this straightforward algorithm runs in polynomial time, that is, that the number of basic steps the algorithm carries out is bounded by a polynomial in $|\alpha|$ where $|\alpha|$ is the number of symbols required to represent α .

First, we argue the number of symbols the algorithm prints, is bounded by a polynomial in $|\alpha|$. In addition to printing the two marker symbols \langle, \rangle and $n - 1$ copies of the comma, the algorithm prints n copies of β_i , that is, n copies of

$$((\bar{x}_i \cdot (\alpha + \hat{\alpha})) + (x_i \cdot \overline{(\alpha + \hat{\alpha})})).$$

For any reasonable representation of β_i , there will be the case that the number of symbols required to represent x_i is bounded by $|\alpha|$; the number of symbols required to represent \bar{x}_i is bounded by $2|\alpha|$; the number of symbols required to represent $\hat{\alpha}$ is

⁶If there are other variables occurring in α , the algorithm can simply rename the variables such that no other variables than x_1, \dots, x_n occur in α .

bounded by $2|\alpha|$. Thus, there exist fixed numbers k and ℓ such that the number of symbols required to represent β_i is bounded by $k|\alpha| + \ell$, and the number of symbols the algorithm prints will be bounded by $2 + (n - 1) + n(k|\alpha| + \ell)$. Now, n is the number of (different) variables in α , and there cannot be more variables in α than there are symbols in α , hence, we also have $n \leq |\alpha|$, and the number of symbols printed will be bounded by

$$2 + (n - 1) + n(k|\alpha| + \ell) \leq 2 + |\alpha| + |\alpha|(k|\alpha| + \ell).$$

Let $p(x) = 2 + x + x(kx + \ell)$ and $p(|\alpha|)$ is a bound on the number of symbols printed by the algorithm.

We want to argue that the algorithm runs in polynomial time, and there is some counting going on before the algorithm starts printing, but there will definitely be a polynomial q such that the number of steps required to complete this counting is bounded by $q(|\alpha|)$. Now, how many step will be required to write $p(|\alpha|)$ symbols? Well, no matter how we count the steps, and no matter how we implement the algorithm, as long as we avoid doing something outright strange or abnormal, there will exist fixed numbers a and b such that the overall number of steps executed is bounded by $q(|\alpha|) + (p(|\alpha|) + a)^b$. Thus, the algorithm runs in polynomial time. This completes the proof the theorem. \square

The proof of the next theorem is analogous to the proof of the previous one. We will give a more laconic proof this time, and it might be helpful for those of readers who are less proficient in complexity theory, to study the proof of Theorem 3 thoroughly before proceeding to the proof of Theorem 4.

Theorem 4 *SAT is polynomial time reducible to coINJ.*

Proof. The input to **SAT** is a boolean expression whereas the input to **coINJ** is an operator expression. We will provide polynomial time computable reduction function f transforming a boolean expression α into an operator expression $e = f(\alpha)$ such that

- if α is satisfiable, then there exist $b, b' \in \mathbb{B}^n$ such that $\llbracket e \rrbracket(b) \neq \llbracket e \rrbracket(b')$ and $b \neq b'$.
- if α is not satisfiable, then $\llbracket e \rrbracket(b) = \llbracket e \rrbracket(b')$ whenever $b \neq b'$.

We can without loss of generality assume that the variables occurring in α are x_1, \dots, x_n . Let 0^n be the bit string of length n where every bit is 0, and let the operator $\text{OP} : \mathbb{B}^n \rightarrow \mathbb{B}^n$ be such that

$$\text{OP}_\alpha(b) = \begin{cases} 0^n & \text{if } \alpha = 1 \text{ under either} \\ & \text{the assignment } x_1 := b_1, \dots, x_n := b_n \\ & \text{or the assignment } x_1 := \bar{b}_1, \dots, x_n := \bar{b}_n \\ b & \text{otherwise.} \end{cases}$$

In the case when α is satisfiable we have $\text{OP}_\alpha(b) = 0^n$ and $\text{OP}_\alpha(\text{INV}(b)) = 0^n$ for some $b \in \mathbb{B}^n$, and hence, since $b \neq \text{INV}(b)$, there exist $b, b' \in \mathbb{B}^n$ such that $\text{OP}_\alpha(b) = \text{OP}_\alpha(b')$ and $b \neq b'$. In the case when α is no satisfiable we have $\text{OP}_\alpha(b) = b$ for any $b \in \mathbb{B}^n$, and hence, $\text{OP}_\alpha(b) \neq \text{OP}_\alpha(b')$ whenever $b \neq b'$. This shows that it is sufficient to define f such that $\llbracket f(\alpha) \rrbracket = \text{OP}_\alpha$.

The definition of the reduction function f . Let $\hat{\alpha}$ denote the boolean expression α

where each occurrence of x_i is replaced by \bar{x}_i . Let $\beta_i \equiv (x_i \cdot \overline{(\alpha + \hat{\alpha})})$ and finally let $f(\alpha) = \langle \beta_1, \dots, \beta_n \rangle$. This completes the definition of f .

We have $\llbracket f(\alpha) \rrbracket = \text{OP}_\alpha$ since

$$\beta_i = \begin{cases} 0 & \text{if } \alpha = 1 \text{ under either} \\ & \text{the assignment } x_1 := b_1, \dots, x_n := b_n \\ & \text{or the assignment } x_1 := \bar{b}_1, \dots, x_n := \bar{b}_n \\ x_i & \text{otherwise.} \end{cases}$$

The proof that f is computable in polynomial time is similar to the corresponding part of the proof of the preceding theorem. $\square\square$

Corollary 1 (i) If **SAT** does not belong to P , then **IDM** does not belong to P . (ii) If **SAT** does not belong to P , then **IDM** does not belong to P .

Proof. We prove (i). First we observe that

$$\mathbf{IDM} \text{ belongs to } P \text{ if and only if its } \mathbf{coIDM} \text{ belongs to } P. \quad (*)$$

This is completely trivial. If you have an algorithm solving a problem in polynomial time, then it is very easy to construct an algorithm solving the co-problem in polynomial time. The same algorithm answering NO in place of YES, and YES in place of NO, will do the job.

Theorem 3 states that **SAT** is polynomial time reducible to **coIDM**. By Theorem 1, it follows that if **SAT** does not belong to P , then neither will **coIDM** do so. Now, (i) follows by (*).

The proof of (ii) is of course similar to the proof of (i). $\square\square$

So far our discussion shows that the problems **IDM** and **INJ** are most likely to not be in P . In the next sections will undertake a further complexity theoretic analysis and see if we can say something more about the inherent computational complexity of these problems.

5 More complexity theory

Complexity can be measured in terms of both time and space resources.

5.1 The space classes LINSPEACE and PSPACE

The definitions of the complexity classes LINSPEACE and PSPACE are analogous to the definition of the class P , but in contrast to imposing a bound on the number of steps executed by an algorithm, we impose a bound on the storage resources required to execute an algorithm.

Definition 7 (The class LINSPEACE and the class PSPACE) Let f be any function from \mathbb{N} to \mathbb{N} and let $|x|$ denote the number of alphabet symbols required to represent the input x . An algorithm taking x as input runs, or works, in space $f(|x|)$ if $f(|x|)$ bounds the number of storage units required to execute the algorithm.

An algorithm works in polynomial space if there exists a polynomial p such that the algorithm works in space $p(|x|)$. A problem belongs to the class PSPACE if (and only if) the problem can be solved by an algorithm working in polynomial space. Further, an algorithm works in linear space if there exists a fixed number k such that the algorithm works in space $k|x|$.

We use standard terminology when we say that the problems in LINSPEACE, respectively PSPACE, are the problems solvable, or decidable, in linear, respectively polynomial, space .

When we defined P , we relied on our intuitive notion of an *algorithm* and, in order to impose a resource bounds on the algorithms, we resorted to another intuitive, but closely related, notion: *the number of basic steps in an execution*. In the definitions of LINSPEACE and PSPACE, we are again relying on our intuitive notion of an algorithm, but now the resource bounds are stated in terms of another informal notion: *the number of storage units required by an execution*. After we defined P , we had a long discussion concluding that P is robust class, and no matter how we define *algorithm* and *basic step*, as long as we are reasonable, the class remains the same. The class P is simply not sensitive to the details in the formal definitions of *algorithm* and *basic step*, nor is the class very sensitive to the conventions for representing inputs. This robustness of P can hardly be overestimated and shows that class P is a natural entity well suited for mathematical studies. Fortunately, both LINSPEACE and PSPACE are in the same boat. Neither of these classes turns out to be sensitive to various details in the definitions, in particular, they are not very sensitive to how define an *algorithm* and the related notion of *the number of storage units required by an execution*.

Hence, if we view an algorithm as a program written in an object oriented programming language, and the number of storage units required as the number of objects created during the execution of the program. However, be reasonable: if we take such a view, we naturally cannot permit the program to store data other places than in the objects, e.g. on files.

Personally we tend to think of an algorithm as a clerical routine carried out by hand. We imagine a clerk writing and erasing symbols in an infinite spreadsheet where each cell have room for exactly one symbol. The symbols should be take from a finite alphabet, but all sorts of symbols can be there (digits, Latin letters, marker symbols, Greek letters, etc.). The number of storage units required is of course the number of cells the clerk have to touch.

In a standard textbook one finds complexity-theoretic spaces classes defined by Turing machines. A Turing machine has an infinite tape that is divided into cells, and the number of storage units required for a computation will be defined as the number of cells the Turing machine's scanning head will visit during a run. Such a formal definition gives associations to the clerk manipulating symbols in a spreadsheet: the clerk is replaced by a mechanical device, and the two-dimensional spreadsheet is replaced by a one-dimensional tape.

5.2 Example: SAT is in LINSPEACE

Let us gain some familiarity with the newly introduced space classes by verifying that the problem **SAT** belongs to LINSPEACE. We will discuss how the clerk

mentioned above, can solve the problem in linear space. The input is a boolean expression, e.g.

$$(x_3 \cdot \overline{(x_{17} + x_2)}) . \quad (16)$$

Now, the clerk needs to write down the input in his spreadsheet, and there is only room for one symbol in each cell, thus, he cannot use subscript or overline groups of symbols. We have to come up with a suitable convention for representing the input, and we can e.g. represent the boolean expression in (16) by

$$(\{3\} . c((\{17\} + \{2\}))) . \quad (17)$$

This is a straightforward and obvious convention where we represent the complement $\overline{\alpha}$ of a boolean expression α by $c(\alpha)$ and a variable by its subscript enclosed in braces. Any reasonable convention will do, and we will stick to this one.

The clerk will write down the input as a sequence of symbols. Immediately after the input, he will write a marker symbol !, and immediately after the marker, for each variable occurring in the input, he will write a copy of the symbol 0. When he is done, we have the configuration

$$(\{3\} . c((\{17\} + \{2\}))) ! 0 0 0 . \quad (18)$$

One can see how the clerk might proceed to solve the problem in linear space. The symbols at the right hand side of the marker can be interpreted as three boolean values, and the clerk will assign these values to the three boolean variables in the input expression. The assignment can e.g. be carried out by writing the values in suitable cells below the expression:

$$\begin{array}{ccccccc} (\{3\} . c((\{17\} + \{2\}))) ! 0 0 0 \\ 0 \qquad \qquad 0 \qquad \qquad 0 \end{array}$$

Now the clerk can proceed in a fairly obvious way and compute the value of the whole expression by computing values of subexpressions. Under the given assignment, the value of $(\{17\} + \{2\})$ is 0, and the value will be written in a suitable cell, e.g. in the cell right below the cell where the subexpression starts:

$$\begin{array}{ccccccc} (\{3\} . c((\{17\} + \{2\}))) ! 0 0 0 \\ 0 \qquad \quad 0 0 \qquad \quad 0 \end{array}$$

Next, the clerk computes the value of $c((\{17\} + \{2\}))$ by reading off the value of $(\{17\} + \{2\})$, the result is e.g. placed right below the c :

$$\begin{array}{ccccccc} (\{3\} . c((\{17\} + \{2\}))) ! 0 0 0 \\ 0 \qquad 1 \ 0 \ 0 \qquad \quad 0 \end{array}$$

And thus the clerk proceeds until he ends up with value of the entire expression. If this value is 1, the clerk's job is done, and he can answer YES (the boolean expression is satisfiable); if the value is 0, he will erase the bottom line, and start all over again from a configuration like (18), but of course, with a new assignment. The clerk will check the assignment

$$(\{3\} . c((\{17\} + \{2\}))) ! 0 0 1$$

Next he checks the assignment

$$(\{3\} . c((\{17\} + \{2\}))) ! 0 1 0$$

and so on. He is instructed to view the bit string at the right hand side of the marker as a binary number, and he gets the subsequent assignments by increasing this number by 1. If he encounters an assignment satisfying the expression, he will quit and say YES (the expression is satisfiable), if not, he will go on until he has checked the assignment 1 1 1, and then he will quit and say NO (the expression is not satisfiable).

It should be fairly obvious from our discussion that the number of cells the clerk will touch, is bounded by a multiple in the number of cells required to represent the input. The bit sequence to the right of the marker will be shorter than the input to the left, and the bit written beneath the input will not require as much space as the input itself. Hence, if $|x|$ is the length of input, the clerk will not touch more than $2|x| + 1$ cells? Well, actually he might touch a few more. We should be careful. This clerk is a strange being. On the one hand, he is extremely unknowledgeable and forgetful, on the other hand, he is a devil at following unambiguous rules and carry them out the letter.⁷ We cannot just tell him to count the number of variables in the input, or just tell him to increase a binary number by 1. We have to give him instructions of how to do so, and carrying out these instruction the clerk will need some auxiliary space. Exactly how much space he need, will of course depend on the instructions; maybe he needs to write down a few marker symbols, maybe he needs to write down some numbers, anyway, there is definitely possible to provide instructions such that the number of auxiliary cells required is bounded by e.g. $2|x| + 1000$. Further, if we just claim *the existence* of fixed numbers k and ℓ such that number of cells required for the entire computation is bounded by $k|x| + \ell$, then we should be on the safe side. But then, the number of cells required is bounded $(k + \ell)|x|$ where $k + \ell$ is a fixed number.⁸ Hence, **SAT** is in LINSPEC.

5.3 Non-Determinism and the class NP

All the algorithms we have encountered so far have been *deterministic algorithms*. We have just called them algorithms since the standard and natural notion of an algorithm is the notion of a deterministic algorithm.

What is a *non-deterministic algorithm*? The very idea of a non-deterministic algorithm seems suspicious and might even sound like a contradiction in terms. Algorithms are supposed to be exhaustive and unambiguous recipes for carrying out tasks mechanically, whereas non-determinism alludes to something arbitrary and non-mechanical.⁹

Now, whereas a deterministic algorithm is a recipe that always tell you what to do next, a non-deterministic algorithm is a recipe that leaves you with several options. Extend a standard programming language, like C or a Java, by the construction

if ? then program 1 **else** program 2

and you have a language capable of expressing any non-deterministic algorithm. When this if-then-else construction is executed, one, and only one, of the two program will be executed, but we do not know which one. The clerk can carry out a non-deterministic algorithm in his spreadsheet the way as he carries out a deter-

⁷Someone might call him a true bureaucrat, but beware, in contrast to a bureaucrat, the clerk is not inclined to act inconsistently.

⁸We assume that the input always has some length, i.e. that $|x| > 0$.

⁹Certain people seem inevitably attracted to a word like “non-determinism”, like they seem attracted to words like “non-reductionism”, “non-monotone” and “non-linear”. Perhaps because such words give associations to something elevated and sophisticated as opposed to their counterparts “monotone”, “linear” and “determinism”.

ministic one, but occasionally he will find his instructions ambiguous in the sense that he could either do A or B , and to decide what to do, he flips a coin.

An algorithm solves a problem if (and only if) the algorithm answer the question correctly, that is, if the answer to the problem on input x is YES, respectively NO, the algorithm should terminate on input x and say YES, respectively NO. Now, a non-deterministic algorithm has several possible executions on one and the same input, and one of the executions on input x might result in a YES whereas another results in a NO. Thus, it is not obvious that it makes any sense saying that a non-deterministic algorithms solves a problems, neither is it obvious that it makes sense to say that a non-deterministic algorithm works in polynomial time, and the reader should study the next definition carefully.

Definition 8 (The class NP) A non-deterministic algorithm solves a problem *if (and only if)*

1. *if the answer to the question on input x is NO, then any execution of the algorithm on input x will give output NO*
2. *if the answer to the question on input x is YES, then there exists an execution of the algorithm on input x giving output YES.*

Let f be any function from \mathbb{N} to \mathbb{N} and let $|x|$ denote the number of alphabet symbols required to represent the input x . A non-deterministic algorithm taking x as input runs, or works, in time $f(|x|)$ if $f(|x|)$ bounds the number of basic steps in any execution of the algorithm on input x .

We will say that a non-deterministic algorithm works in polynomial time if there exists a polynomial p such that the non-deterministic algorithm works in time $p(|x|)$.

A problem belongs to the class NP if (and only if) the problem can be solved by an non-deterministic algorithm working in polynomial time. We use standard terminology when say that the problems in NP are the problems solvable, or decidable, in non-deterministic polynomial time.

It should be obvious to any proficient computer programmer that any problem solved by a non-deterministic algorithm also can be solved by a deterministic one. On a particular input x there will be finitely many executions of the non-deterministic algorithm, and by applying a standard technique called back-tracking, a deterministic algorithm can systematically check out each and one of the executions. If at least one of the executions results in a YES, the answer to the problem of input x is also YES, and if every execution results in a NO, the answer is NO.

Hence, if non-deterministic algorithm solves a problem, there will also be a deterministic algorithm solving the problem. So what is all the fuss about? If non-determinism adds no computational power, why do we bother to introduce non-determinism at all? This is why: If a non-deterministic algorithm solves a *problem in polynomial time*, there will not necessarily be a deterministic algorithm solving the problem *problem in polynomial time*.

We can always simulate a non-deterministic algorithm by a deterministic one, but to simulate a non-deterministic algorithm running in polynomial time, will in general require exponential time. It is very likely that the class of problems solvable in deterministic polynomial time, that is P, is strictly included in the solvable in non-deterministic polynomial time, that is NP, but as mentioned above, this is one of the most famous open problems of mathematics.

5.4 Example: SPLIT and SAT are in NP

In Section 3.1 we discussed the problem **SPLIT**, and we rhetorically ask for the solution of the problem on the input

$$\{3, 6, 9, 10, 20, 30, 173, 174, 175, 1027, 1058, 1132, 1719, 2480\}. \quad (19)$$

Is it possible to split the numbers in the set into two piles such that the sum of the numbers in one pile equals the the sum of the numbers in the other? We ask this question to make the reader realise that **SPLIT** is an intuitively hard problem. The reader that tried to answer the question, probably got bored after a while because he had to check out various ways to split the set, and there are awfully many ways to split the set, 2^{14} ways to be exact, that is more than 15 thousand possibilities. Maybe the reader did not find it necessary to check all the 2^{14} splittings one by one, but he probably realised that he would be forced to check quite a few of them, and then he eventually gave up. No one has ever found a fast deterministic algorithm for solving the problem **SPLIT**, and the problem seems intractable. However, it is easy to come up with a non-deterministic procedure solving the problem very quickly:

```

Input: a set  $X$  of natural numbers.
foreach  $a \in X$  do
    if ? then put  $a$  in  $Y$  else put  $a$  in  $Z$ 
 $y :=$  the sum of the numbers in  $Y$ 
 $z :=$  the sum of the numbers in  $Z$ 
if  $x = y$  then SAY YES else SAY NO

```

Given the input in (19), the algorithm will run through the numbers

$$3, 6, 9, 10, 20, 30, 173, 174, 175, 1027, 1058, 1132, 1719, 2480$$

and non-deterministically put some of them in Y and some of them in Z . It is possible that 30, 174, 1027, 1058, 1719 end up in Y , and that 3, 6, 9, 10, 20, 173, 175, 1132, 2480 end up in Z . Then the output will be YES since

$$\begin{aligned} 30 + 174 + 1027 + 1058 + 1719 &= 4008 = \\ 3 + 6 + 9 + 10 + 20 + 173 + 174 + 175 + 1027 + 1058 + 1132 + 1719 + 2480. \end{aligned}$$

It is also e.g. possible that 3 and 9 ends up in Y and the rest of the numbers in Z . Then the output will be NO.

When we recall the definition of what it means for a non-deterministic algorithm to solve a problem, it is obvious that the given algorithm solves **SPLIT**. The algorithm is very fast and it is easy to argue that any execution of the algorithm runs in polynomial time in the length of the input. Hence the problem **SPLIT** is NP.

The problem **SAT** is also in NP. In Section 5.2 we argued that **SAT** is in LINSPEACE by showing how a clerk with a spreadsheet could solve the problem in linear space. The clerk was instructed to systematically run through all possible assignments and check if one of them satisfied the boolean expression given as input. Now, if we in contrast to instruct the clerk to check all assignments, instruct him to non-deterministically generate a single assignment and check that particular assignment, then we have a non-deterministic algorithm solving **SAT**. (If the assignments he is led to by flipping a coin, satisfies the input expression, he should say YES, otherwise, he should say NO.) But then, just checking a sole assignment, the clerk will be done very quick, and it is easy to see that the number of basic steps he has to carry out is bounded by a polynomial in the length of the input. Hence, the problem **SAT** is in NP.

5.5 The class co-NP

Definition 9 Let $COMP$ be any complexity class (e.g. P , NP , $LINSPACE$). We define the complexity class $co-COMP$ by

coA is in $co-COMP$ if and only if A is in $COMP$.

If we have a deterministic algorithm solving a problem A , we can trivially modify the algorithm to solve the problem coA , just modify the algorithm to give output NO in place of YES, and output YES in place of NO. The modified algorithm requires exactly the same amount of time and space resources as the initial algorithm, and thus the problem coA will be in exactly the same deterministic complexity classes as the problem A . Thus, e.g. $P = co-P$ and $LINSPACE = co-LINSPACE$ since P and $LINSPACE$ are deterministic complexity classes, that is, they are complexity classes defined by the resource requirements of deterministic algorithms.

Recall the definition of what it means for a non-deterministic algorithm to solve a problem: An non-deterministic algorithm solves a problem if and only if

1. if the answer to the question on input x is NO, then any execution of the algorithm on input x will give output NO
2. if the answer to the question on input x is YES, then there exists an execution of the algorithm on input x giving output YES.

Notice the asymmetry: The answer to the problem is YES when *one* execution gives output YES (even if many other executions give output NO); the answer to the problem is NO when *all* executions give output NO (and not a single execution gives the output YES). This asymmetry implies that we cannot modify a non-deterministic algorithm solving a problem A to solve the problem $co-A$ by swapping its outputs YES and NO, which again implies that the complexity class NP probably is different from the complexity class $co-NP$.

To determine the relationship between NP and $co-NP$ is one of the famous open problems of complexity theory. It is likely that the two classes are different, but the research community will perhaps not be as unanimous in this respect as with respect to the problem $P \stackrel{?}{=} NP$. Since $P = co-P$, it cannot be the case that we both have $NP \neq co-NP$ and $P = NP$. Thus, to prove that NP is different from $co-NP$, is a way of proving that P is different from NP . (However, if NP should equal $co-NP$, it might still be the case that P is different from NP .)

5.6 More on the relationship between the complexity classes

It is proven (see fig. 7) that

$$P \subseteq NP \subseteq PSPACE. \quad (20)$$

It is an open problem whether $P \neq PSPACE$, and hence, it is also an open problem if $P \neq NP$ and if $NP \neq PSPACE$. The problems are considered open as no one has come up with rigorous mathematical proofs of these inequalities, however, it is considered highly likely that all the inequalities hold, and we have absolutely no experience indicating otherwise. Thus, the likely situation is that all the inclusions in 20 are strict, that we indeed have

$$P \subset NP \subset PSPACE. \quad (21)$$

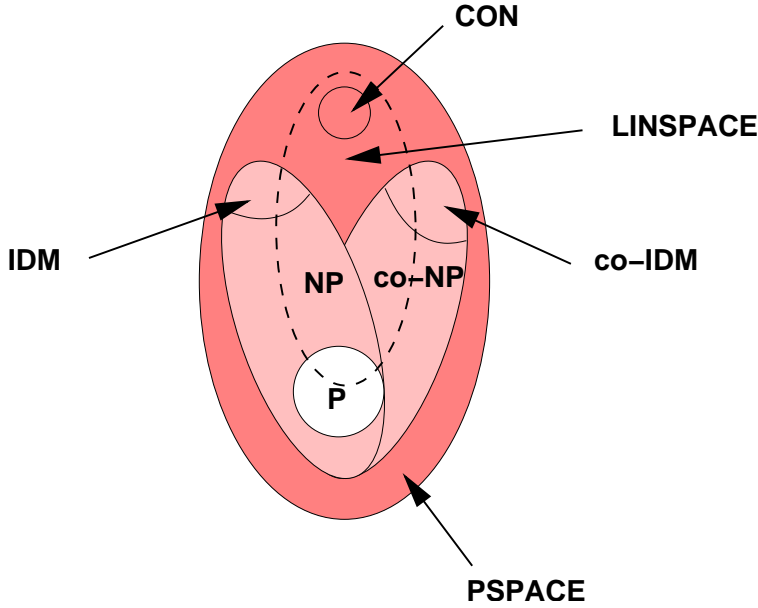


Figure 7: Locations of problems in the space classes.

Now, if we believe that 21 holds, we are also bound to believe that

$$P \subset \text{co-NP} \subset \text{PSPACE} . \quad (22)$$

Why are we bound to believe that (22) holds? Well, PSPACE is a deterministic complexity class, and hence, if a problem A belongs to PSPACE, the co-problem $\text{co}A$ does also belong to PSPACE. Hence, we are bound to believe that co-NP is included in PSPACE. Further, we cannot believe that $\text{co-NP} = \text{PSPACE}$ since this entails that $\text{NP} = \text{PSPACE}$ (and we don't believe that). Hence we have to believe that co-NP is strictly included in PSPACE. A symmetric argument shows that we also are bound to believe that P is strictly included in co-NP .

The relationship between LINSPEC and the other complexity classes is slightly complicated. It is proved that $\text{LINSPEC} \subset \text{PSPACE}$. Further, it is proved that LINSPEC is different from P , that is $\text{LINSPEC} \neq P$. Thus, it follows that we have one of the following situations

1. $\text{LINSPEC} \subset P$ (LINSPEC is strictly included in P)
2. $P \subset \text{LINSPEC}$ (P is strictly included in LINSPEC)
3. $P \not\subseteq \text{LINSPEC}$ and $\text{LINSPEC} \not\subseteq P$ (neither of the classes is contained in the other)

Any of three situations is compatible with what we can prove, namely that LINSPEC cannot equal P , however, Situation (1) implies that $P = \text{PSPACE}$ and we don't believe that, so we have to rule out (1). (The argument (1) implies that $P = \text{PSPACE}$ is nontrivial, and we omit the details.) Situation (2) does not seem to be incompatible with anything else we believe in, but the situation is still considered unlikely, and the general opinion among complexity theorists is that Situation (3) is realised.

It is also proved that $\text{LINSPEC} \neq \text{NP}$, and the likely situation is that neither of classes is included in the other, that is, $\text{NP} \not\subseteq \text{LINSPEC}$ and $\text{LINSPEC} \not\subseteq \text{NP}$. The same can be said about the relationship between LINSPEC and co-NP .

5.7 The importance of the complexity classes

The complexity class P delineates a border between the tractable and the intractable. If a problem belongs to P , real-life computing machinery can predictably solve the problem, if a problem does not belong to P , such machinery cannot deal with the problem. (See Section 3.2.) This makes P an important complexity class. To realise why NP is an important complexity class, we need the next definition.

Definition 10 *A problem A is NP-hard if (and only if) if every problem in NP is polynomial time reducible to A . Further, a problem A is NP-complete if (and only if)*

1. A is NP-hard
2. A belongs to the class NP

The NP-complete problems are the hardest problems in the class NP , and by studying the definition the reader will realise that if a single NP-complete problem can be solved in polynomial time, then every problem in NP can be solved in polynomial time. Now, since it is very likely that $P \neq NP$, it is also very likely that no polynomial-time algorithm can solve an NP-complete problem, and it is not only very like, but for sure absolutely true, that no one in the history of computing has come upon a polynomial-time algorithm solving an NP-complete problem. Thus, NP-complete problems are intractable problems, that is, problems computing machinery cannot deal with. Besides, numerous natural problems from engineering, mathematics, science and daily life have turned out to be NP-complete, and thus NP becomes a very important complexity class. By proving that a problem is NP-complete, and a lot of natural problems will be NP-complete, we have proved that the problem is intractable. This is for sure a useful thing to know about a problem.

The notion of PSPACE-completeness is defined analogous to the notion of NP-completeness.

Definition 11 *A problem A is PSPACE-hard if (and only if) if every problem in PSPACE is polynomial time reducible to A . Further, a problem A is PSPACE-complete if (and only if)*

1. A is PSPACE-hard
2. A belongs to the class PSPACE

The PSPACE-complete problems are the hardest problems in PSPACE, and they should be even harder to solve than NP-complete problems. As far as we know $NP \neq PSPACE$, and then even a *non-deterministic* algorithm will require *exponential* time to solve a PSPACE-complete problem.

Computers can to a certain extent deal with NP-complete problems. They cannot deal with such problems in their full generality, but techniques and methods have been developed to find approximate and partial solutions. If a programmer realises he is facing an NP-complete, the situation is for sure not ideal, but the situation is not hopeless as he might be able to make his software respond satisfactorily on normal inputs. (There is a vast theory on how programmers can approach intractable problems.) If a programmer realises he is facing an PSPACE-complete, he is worse off, and it would probably be far more difficult to make the software behave satisfactorily.¹⁰

¹⁰An important mission of theoretical computer science is to tell engineers, programmers and applied computer scientists what they *cannot do*. This might be more important than telling them

There are not nearly as many PSPACE-complete natural problems as there are NP-complete natural problems, but a significant number of interesting natural problems have turned out to be PSPACE-complete, and PSPACE is an important complexity class.

The class co-NP gains its importance from being the co-class of NP. What should we say about LINSPEACE? In what sense is LINSPEACE an important complexity class? Well, the class is surrounded by other important complexity classes, and that makes the class interesting, particularly since the relationship between LINSPEACE and these classes is not settled. Besides, it is of course nice to know that it is possible to solve a problem in very limited space, still, from an applied point of view, LINSPEACE might not be as important as P, NP and PSPACE. (Note that LINSPEACE is not closed under polynomial time reductions, that is, a problem lying outside LINSPEACE might be polynomial time reducible a problem in LINSPEACE.)

In the literature the reader might find numerous complexity classes that we have not found time to discuss in this paper, but the ones we have discussed, are all very important and well-known complexity classes. The most prominent class that we have ignored in this is called LOGSPACE. (It is proved that $\text{LOGSPACE} \subseteq \text{P}$, and it is an open problem if $\text{LOGSPACE} \neq \text{P}$.) Other fairly well-known classes are known as NC , NLOGSPACE , NLINSPEACE , ETIME , NETIME and $\text{P}^\#$.

6 More on operators and configuration management

6.1 More on the complexity of IDM and INJ

Some ideas needed to prove Theorem 5 and Theorem 6, are explained in Section 5.2, and the reader might benefit from studying the details of the section thoroughly before he embarks on the proofs of the theorems.

Theorem 5 *The problem INJ is in LINSPEACE.*

Proof. In Section 5.2 we showed how a clerk calculating in a spreadsheet could solve the problem **SAT** in linear space. The input to the problem was a boolean expression α , and the clerk generated systematically all possible assignments to the variables of α , and for each assignment, the clerk evaluated α .

Now, the input to **INJ** is an operator expression e , and an operator expression is nothing but a sequence $\langle \alpha_1, \dots, \alpha_n \rangle$ where each α_i is a boolean formula over the variables x_1, \dots, x_n . We can argue as in Section 5.2 that a clerk working in linear space, can assign values to x_1, \dots, x_n , and for each assignment, he evaluate the expressions $\alpha_1, \dots, \alpha_n$. The evaluation of $\alpha_1, \dots, \alpha_n$ under a particular assignment yields a string of n bits, and this string is of course nothing but the result of applying the operator $\llbracket e \rrbracket$ to a particular argument in \mathbb{B}^n . Thus, it should be clear that a

what they *can do* since they might know a lot about that already.

clerk can carry out the following routine in linear space:

```

Input: an operator expression  $e$ .
 $n :=$  the number of variables in  $e$ 
Let  $b^0, b^1, \dots, b^{2^n-1}$  be the obvious enumeration of the elements in  $\mathbb{B}^n$ 
for  $i := 0 \dots 2^n - 2$  do
  begin
     $b := \llbracket e \rrbracket(b^i)$ 
    for  $j := i + 1 \dots 2^n - 1$  do
      begin
         $c := \llbracket e \rrbracket(b^j)$ 
        if  $b = c$  then SAY NO AND QUIT else PROCEED
      end
    end
  end
SAY YES

```

The routine is simply a brute force check. For each bit string b^i , the clerk check if $\llbracket e \rrbracket(b^i) = \llbracket e \rrbracket(b^j)$ for some bit string b^j where $j > i$. (Obviously, there is no need check when $j < i$, but it would not have harmed if we did so.) If the clerk finds two bit strings witnessing that the operator given by the input is not an injection, he says NO and quits the routine. If such witnesses never turn up, the clerk says YES after the checking is done.

This proves that **INJ** is in LINSPEC. $\square\square$

Theorem 6 *The problem **IDM** is in LINSPEC.*

Proof. This proof is similar to the proof of Theorem 5, but slightly simpler. Again, we give a routine meant to be carried out by a clerk working in a spreadsheet:

```

Input: an operator expression  $e$ .
 $n :=$  the number of variables in  $e$ 
Let  $b^0, b^1, \dots, b^{2^n-1}$  be the obvious enumeration of the elements in  $\mathbb{B}^n$ 
for  $i := 0 \dots 2^n - 1$  do
  begin
     $c := \llbracket e \rrbracket(b^i)$ 
     $d := \llbracket e \rrbracket(c)$ 
    if  $c = d$  then PROCEED else SAY NO AND QUIT
  end
SAY YES

```

The clerk is instructed to check for absolutely every bit string $b \in \mathbb{B}^n$ and see if the string yields a counterexample to the equation $\llbracket e \rrbracket(b) = \llbracket e \rrbracket(\llbracket e \rrbracket(b))$. If a counterexample turns up, the clerk will interrupt the routine and give output NO, and if no counterexample turns up, he will give output YES. Following the lines of the reasoning in Section 5.2, the reader can verify that it is possible for the clerk to carry out this routine in linear space. Thus, the problem **IDM** is in LINSPEC. $\square\square$

Theorem 7 *The problem **coINJ** is NP-complete.*

Proof. We are going to prove that **coINJ** is NP-complete. How should we proceed? Well, we have to consult our definitions to find out what it means that the problem **coINJ** is NP-complete. Definition 10 states that

$coINJ$ is NP-complete if and only if (1) $coINJ$ is NP-hard and (2) $coINJ$ belongs to the class NP.

Thus, we have to prove two things, i.e. we have to prove that (1) $coINJ$ is NP-hard, and we have to prove that (2) $coINJ$ belongs to the class NP.

First we prove (1). According to the definition of NP-hardness (Definition 10) we have to prove that every problem in NP is polynomial time reducible to $coINJ$. In the proof we will apply the the following claim.

(Claim) The problem **SAT** is NP-hard.

A proof of (Claim) can be found in any standard textbook on basic complexity theory.

Pick an arbitrary problem A in the class NP. Since we know that **SAT** is NP-hard, we know that A is polynomial time reducible to **SAT**, and hence, according to Definition 3 is there exists g such that

- The answer to question A with input x is the same as the answer to question **SAT** with input $g(x)$
- The function g is computable in polynomial time.

Further, Theorem 3 states that the problem **SAT** is polynomial time reducible to $coINJ$, and hence, again according to Definition 3 is there exists h such that

- The answer to question **SAT** with input x is the same as the answer to question $coINJ$ with input $h(x)$
- the function h is computable in polynomial time.

Now, let $f(x) = h(g(x))$. The reader can easily check that

- The answer to question A with input x is the same as the answer to question $coINJ$ with input $f(x)$
- The function f is computable in polynomial time.

Hence, again by Definition 3, the problem A is polynomial time reducible to $coINJ$. Now, A was arbitrarily chosen from the class NP, and we have not assumed that the problem possess any particular properties beyond belonging to NP. Hence, any problem in NP will be polynomial time reducible to $coINJ$ which means that the problem $coINJ$ is NP-hard.

We will now turn to the proof of (2), that is, we will prove that $coINJ$ belongs to NP, and we will do so by providing an non-deterministic algorithm solving $coINJ$ in polynomial time. If such an algorithm exists, the definition of NP says that $coINJ$ is in NP. Let us remind ourself of how the problem $coINJ$ is stated.

($coINJ$) *Input:* an operator expression e .

Question: Do we have $\llbracket e \rrbracket(b) = \llbracket e \rrbracket(b')$ and $b \neq b'$ for some $b, b' \in \mathbb{B}^n$?

It is obvious that whenever an operator expression e and two bit string $b, b' \in \mathbb{B}^n$ are available, then an deterministic algorithm can check quite fast if it will be the case that $\llbracket e \rrbracket(b) = \llbracket e \rrbracket(b')$ and $b \neq b'$, and there is beyond any doubt that such an algorithm will work in polynomial time in the length of the input expression

e . Now, n , that is the length of bit strings b and b' , will definitely be shorter than the length of the input, and thus, a non-deterministic algorithm working in polynomial (even linear) time in the length of input might non-deterministically generate two candidates for b and b' . Hence, it is easy to see that there exists a non-deterministic polynomial time algorithm solving **coINJ**: simply generate non-deterministically the two bit strings b and b' , and thereafter check if it is the case that $\llbracket e \rrbracket(b) = \llbracket e \rrbracket(b')$ and $b \neq b'$. If it is the case the answer to the problem is YES, else the answer is NO. $\square\square$

Theorem 8 *The problem **coIDM** is NP-complete.*

Proof. The proof of this theorem is analogous to the proof of Theorem 7. We omit the details. $\square\square$

Corollary 2 *(i) The problem **INJ** is in co-NP, furthermore, any problem in co-NP is polynomial time reducible to **INJ**, i.e., **INJ** is co-NP-complete. (ii) The problem **IDM** is in co-NP, furthermore, any problem in co-NP is polynomial time reducible to **INJ**, i.e., **IDM** is co-NP-complete.*

Proof. (i) follows straightforwardly from Theorem 7, and (ii) follows straightforwardly from Theorem 8. $\square\square$

6.2 The Complexity of **CON** and **CONx**

Next we turn to the so-called ‘convergent’ operations, in the classical sense of the phrase. Ideally we want an operator that not only converges, but which converges always to the same value in the fixed-point sense of cfengine[3], i.e. $\llbracket e \rrbracket^k(x) = \llbracket e \rrbracket^{k+1}(x) = \text{const}$ policy for all x . However, this problem is at least as hard as the general convergent case, since we must first check for convergence and then additionally check whether the value is correct, so we do not need to mention it again.

6.2.1 A brief discussion

Note the difference between the problem **CON** and the problem **CONx**. If you are asked to solve the problem **CONx**, you will be given a pair $\langle e, b \rangle$ where $b \in \mathbb{B}^n$ and e is an operator expression of the form $\langle \alpha_1, \dots, \alpha_n \rangle$, and then, it is your task to decide if the operator $\llbracket e \rrbracket$ converges in the particular argument b , that is, you should decide if there exists a k such that $\llbracket e \rrbracket^k(b) = \llbracket e \rrbracket^{k+1}(b)$. If you are asked to solve the problem **CON**, you will be given nothing but an operator expression of e of the form $\langle \alpha_1, \dots, \alpha_n \rangle$, and it is your task to decide if the operator $\llbracket e \rrbracket$ converges for every $x \in \mathbb{B}^n$. This sounds like a formidable task as you have to check if the operator converge for each of the 2^n elements in \mathbb{B}^n . Thus, at a first glance the problem **CON** might seem harder than the problem **CONx**, and it certainly does not seem any easier.

In fact even **CONx** is a hard problem, indeed, the problem turns out to be PSPACE-hard. So far we have done our best to avoid Turing machines, but now we are forced to introduce them as we will prove that **CONx** is PSPACE-hard by constructing operators simulating Turing machines.

6.2.2 Turing Machines

A Turing machine might be seen as a mechanical device shuffling symbols around an infinite tape, which models its addressable memory. The tape is divided into cells, or squares; the tape starts somewhere but does not end anywhere. Thus, we can talk about the first cell of the tape, the second cell of the tape, and in general the n 'th cell of the tape. Each cell has room for exactly one symbol taken from a finite *alphabet*. The alphabet varies from machine to machine, but it is always finite, and it always contains a dedicated symbol called the *blank* and denoted **B**. A cell containing **B** will be called a *blank cell*. At any particular moment in time the machine will be in one, and only one, of a finite number of possible states, and a device called the machine's *head*, will be scanning one, and only one, of the cells on the tape. The symbol in the scanned cell is called the *scanned symbol*. The combination of the current state and the scanned symbol decides what the Turing machine will do next, and the possible actions are indeed very limited:

- (i) write a symbol in the scanned cell and enter a new state
- (ii) move the head one position to the left and enter a new state
- (iii) move the head one position to the right and enter a new state

The exact action to be executed is determined by a finite set of instructions of the form

$$a, q, A, r$$

where a is an alphabet symbol, q and r are states, and A is either an alphabet symbol, or a left arrow (\leftarrow), or a right arrow (\rightarrow). If the machine is in the state q and is scanning the symbol a , then the machine will carry out the action A and enter state r . If A is an alphabet symbol, the action will simply be to write that symbol in the scanned square; if A is an arrow, the action will simply be to move the scanning head one position in the direction of the arrow. If the Turing machine is *deterministic*, there will never be more than one instruction in the set that is applicable in a particular situation, if the Turing machine is *nondeterministic*, several instructions might be applicable, and the machine will execute one of them. It might also be the case that none of the instructions are applicable, then the machine will simply “hang”.

When a Turing machine starts, the leftmost cell on the tape, that is, the first cell, will be blank, and the input will be found, symbol by symbol, in the consecutive cells. The input will not contain any blanks, and the head will be scanning the first symbol of the input. The remaining cells of the infinite tape are all blank. E.g., if the input *abbaa* is given to a machine, the tape will be in the configuration

B	<i>a</i>	<i>b</i>	<i>b</i>	<i>a</i>	<i>a</i>	B	B	B	B	B	B	B	B	B	B	B	...

↑

when the execution starts. The arrow marks the scanned cell.

Turing machines can perform tasks like computing functions and deciding problems by manipulating the symbols on the tape. The instructions tell a machine how to write and delete symbols and how to move the head back and forth. To delete a symbol from a particular cell, does of course amount to write the blank symbol into that cell. When a new symbol is written into a cell, the old symbol vanishes and is lost forever. Let us study a Turing machine copying a string of a 's. The string might be of arbitrary length and is given to the machine as input, so the Turing

- (i) Alphabet: $\{a, X, \mathbf{B}\}$
- (ii) States: $\{q_0, q_1, q_2, q_3, q_4, q_5\}$
- (iii) Instructions:

B	q_1	B	q_0	(inst. 1)	B	q_3	a	q_4	(inst. 7)
a	q_1	X	q_2	(inst. 2)	a	q_4	\leftarrow	q_4	(inst. 8)
X	q_2	\rightarrow	q_2	(inst. 3)	B	q_4	\leftarrow	q_4	(inst. 9)
a	q_2	\rightarrow	q_2	(inst. 4)	X	q_4	a	q_5	(inst. 10)
B	q_2	\rightarrow	q_3	(inst. 5)	a	q_5	\rightarrow	q_1	(inst. 11)
a	q_3	\rightarrow	q_3	(inst. 6)					

- (iv) Start state: q_1

Figure 8: A Turing machine for copying a string of a 's.

machine's task is to bring the tape from the configuration

$$\text{B} \underbrace{a \dots a}_n \text{BBBBBBBBBBBBBB} \dots$$

to the configuration

$$\mathbf{B} \underbrace{a \dots a}_n \mathbf{B} \underbrace{a \dots a}_n \mathbf{B} \mathbf{B} \mathbf{B} \mathbf{B} \mathbf{B} \mathbf{B} \mathbf{B} \mathbf{B} \dots$$

for any $n \in \mathbb{N}$. To define a particular Turing machine, we need to give (i) the alphabet, (ii) the states, (iii) the instructions and (iv) the start state, that is, the state the Turing machine shall be in when the execution starts. The machine copying a string of a 's, is given in Figure 8.

We might visualise the execution of a Turing machine (at a particular input) as a sequence of configurations c_0, c_1, c_2, \dots . A single configuration is a pair consisting of

- a finite sequence of alphabet symbols where exactly one symbol is underlined
- a state.

The state of the configuration gives the current state of the Turing machine, and the sequence of alphabet symbols gives the current content of the tape, symbol by symbol, from the left to the right. If there are n symbols in the sequence, the sequence gives the content of the n leftmost cells of the tape. The remaining cells on the infinite tape will all be blank. The underlined symbol is of course the scanned symbol.

The execution of our copy machine on the input aa results in the sequence of configurations in Figure 9.

We will not spend any time considering the various ways a Turing machine might yield output. We are only interested in deterministic terminating Turing machines solving problems, and the answer to a problem is just YES or NO, and hence, all we need to provide is a convention for Turing machine to say YES/NO. In this respect we follow the standard literature and equip the Turing machine with two dedicated states: the *accept* state q_A and the *reject* state q_R . Furthermore, a Turing machine solving a problem A should starting with the input to A on its tape end up in

Config.:	Tape:	State:	Comments:
c_0	B<u>a</u>aB	q_1	initial config.
c_1	B<u>X</u>aB	q_2	by (inst. 2)
c_2	BX<u>a</u>B	q_2	by (inst. 3)
c_3	BXa<u>B</u>B	q_2	by (inst. 4)
c_4	BXaBB<u>B</u>	q_3	by (inst. 5)
c_5	BXaB<u>a</u>B	q_4	by (inst. 7)
c_6	BXa<u>B</u>aB	q_4	by (inst. 8)
c_7	BX<u>a</u>BaB	q_4	by (inst. 9)
c_8	B<u>X</u>aBaB	q_4	by (inst. 8)
c_9	B<u>a</u>aBaB	q_5	by (inst. 10)
c_{10}	Ba<u>a</u>BaB	q_1	by (inst. 11)
c_{11}	Ba<u>X</u>BaB	q_2	by (inst. 2)
c_{12}	BaX<u>B</u>aB	q_2	by (inst. 3)
c_{13}	BaXB<u>a</u>B	q_3	by (inst. 5)
c_{14}	BaXBa<u>B</u>B	q_3	by (inst. 6)
c_{15}	BaXBa<u>a</u>B	q_4	by (inst. 7)
c_{16}	BaXB<u>a</u>aB	q_4	by (inst. 8)
c_{17}	BaX<u>B</u>aaB	q_4	by (inst. 8)
c_{18}	Ba<u>X</u>BaaB	q_4	by (inst. 9)
c_{19}	Ba<u>a</u>BaaB	q_5	by (inst. 10)
c_{20}	Baa<u>B</u>aaB	q_1	by (inst. 11)
c_{21}	Baa<u>B</u>aaB	q_0	by (inst. 1)

Figure 9: A sequence of configurations generated by the machine copying a string of a 's. The input is the string aa .

- the accept state q_A if the answer to A on input x is YES
- the reject state q_R if the answer to A on input x is NO.

After the machine has entered the accept state or the reject state, no instruction should be applicable, that is, the machine should halt.

6.2.3 Operators Simulating Turing Machines

We can view a deterministic Turing machine M as an operator T_M , that is, T_M is a unary function transforming one configuration into another one, and when M makes a transition from the configuration c to the configuration c' , then T_M makes the same transition, that is, $T_M(c) = c'$. If none of the instructions of M 's is applicable to a given configuration c , we define $T_M(c) = c$. Thus, if M is a machine generating the sequence of configurations given in Figure 9, we have¹¹

$$\begin{aligned}
T_M(\mathbf{B}\underline{a}a\mathbf{B}q_1) &= \mathbf{B}\underline{X}a\mathbf{B}q_2 \\
T_M(\mathbf{B}\underline{X}a\mathbf{B}q_2) &= T_M(T_M(\mathbf{B}\underline{a}a\mathbf{B}q_1)) = \mathbf{B}X\underline{a}\mathbf{B}q_2 \\
T_M(\mathbf{B}X\underline{a}\mathbf{B}q_2) &= T_M(T_M(\mathbf{B}\underline{X}a\mathbf{B}q_2)) = T_M(T_M(T_M(\mathbf{B}\underline{a}a\mathbf{B}q_1))) = \mathbf{B}Xa\underline{\mathbf{B}}\mathbf{B}q_2 \\
&\vdots \\
&\text{etc.}
\end{aligned}$$

We will call such an operator T_M the *transition function* of M . Note that if no instruction of M 's is applicable for the configuration c , we have $T_M(c) = c$ by definition.

Now, suppose the Turing machine M is solving the problem A , that is, when M is executed on input x , then M will halt in

- the accept state q_A if the answer to A on the input x is YES
- the reject q_R if the answer to A on the input x is NO.

Let us modify M slightly such that in contrast to rejecting by entering the state q_R , M rejects by moving the head back and forth between two cells forever and ever. Thus, when M is given the input x , the execution of M will either end up in the state q_A (accepting x) or end up alternating between two configurations $c, c', c, c', c, c', \dots$ (rejecting x). Let T_M be the transition function of this modified version of M , and let c be the initial configuration for M at the input x . Then, if the answer to A on the input x is YES, there exists k such that $T_M^{k+1}(c) = T_M^k(c)$, and if the answer to A on the input x is NO, no such k exists, in other words, T_M converges in the argument c if and only if the answer to A on the input x is YES.

Now, a configuration can of course be represented as a sequence of bits, and if a Turing machine working in polynomial space generates the configurations c_0, c_1, c_2, \dots , the number of bits required to represent each configuration c_i will be bounded by a polynomial in the length of the input, hence, if $n \geq p(|x|)$ where x is the input and p is some fixed polynomial, bit strings taken from \mathbb{B}^n will be large enough to encode any configuration in the sequence c_0, c_1, c_2, \dots . This explains why the next theorem holds.

¹¹We write the state of the configuration immediately after the sequence of alphabet symbols giving the configuration of the tape.

Theorem 9 *Let M be a deterministic Turing machine solving a problem in polynomial space. There exists a fixed polynomial p such that for any input x to M there exist a bit string $b \in \mathbb{B}^n$ and an operator $T: \mathbb{B}^n \rightarrow \mathbb{B}^n$ where $n = p(|x|)$ such that the following assertions are equivalent:*

- *T converges in the argument b*
- *the execution of M on input x halts in the accept state q_A .*

6.2.4 CONx is PSPACE-hard.

To prove that **CONx** is PSPACE-hard, we have to prove that every problem in PSPACE is polynomial time reducible to **CONx**, that is, we have to prove that for every problem A in PSPACE there exists a polynomial time computable function f such that the answer to the problem A on the input x is the same as the answer to the problem **CONx** on the input $f(x)$. (See Definition 11 and Definition 3.)

If a problem A belongs to PSPACE there does of course exist a deterministic Turing machine M solving A in polynomial space. Now, have a second look at Theorem 9. Read the theorem carefully and do particularly note that

...for any input x to M there exist $b \in \mathbb{B}^n$ and $T: \mathbb{B}^n \rightarrow \mathbb{B}^n$...

The operator T is more or less the transition function for the Turing machine M , and the bit string b codes the initial configuration of the execution of M on input x . When x and M are given, it will be possible to construct the bit string b and an operator expression e for the operator T , that is, e such that $\llbracket e \rrbracket = T$. Moreover, the construction can be carried out by an algorithm running in polynomial time in the length of x . (The Turing machine M is fixed and should not be viewed as input to the algorithm.) Any trained complexity theorist can easily design such an algorithm, but since the details are involved and cumbersome, we will save the reader from the entire story. The average reader should be satisfied to realise the consequences of having such an algorithm: There exists a polynomial time computable function f such that when $f(x) = \langle e, b \rangle$, then the answer to the problem A on input x is the same as the the answer to problem **CONx** on input $\langle e, b \rangle$. Hence, A is polynomial time reducible to **CONx**. But A is an arbitrary problem in PSPACE. Hence, any problem in PSPACE is polynomial time reducible to **CONx**, and thus, the next theorem holds.

Theorem 10 *CONx is PSPACE-hard.*

Let us do a brief summary sketching the proof of the theorem once more: Let A be an arbitrary problem in PSPACE. (When we have proved that A is polynomial time reducible to **CONx**, we have proved that **CONx** is PSPACE-hard, see Definition 11.) Let M be a deterministic Turing machine solving the problem A in polynomial space. (Since A is in PSPACE, we know that such a Turing machine exists.) We have a polynomial time computable function f such that when $f(x) = \langle e, b \rangle$, the execution of M on input x halts in the accept state q_A if, and only if, the operator $\llbracket e \rrbracket$ converges in the argument b . Thus, when $f(x) = \langle e, b \rangle$, the following assertions are equivalent:

- the answer to A on input x is YES
- the execution of M on input x halts in the state q_A

- the operator $\llbracket e \rrbracket$ converges in the argument b
- the answer to the problem **CON_x** on input $\langle e, b \rangle$ is YES.

Thus, the answer to the problem A on input x is the same as the answer to the problem **CON_x** on input $f(x)$, and since f is a polynomial time computable function, the problem A is polynomial time reducible to **CON_x**.

6.2.5 Our last words on the complexity of **CON_x** and **CON**

It follows Theorem 10 that if **CON_x** can be solved in polynomial time, then any problem in PSPACE can be solved in polynomial time. Thus, it makes very good sense to say that **CON_x** is at least as hard as any other problem in PSPACE, but exactly how hard is the problem? Is the problem much harder than the other problems in PSPACE, or can the problem itself be solved in polynomial space? Well, it turns out that problem indeed can be solved in linear space and hence belongs to LINSPEACE! This might sound like a contradiction since we know that LINSPEACE is strictly included in PSPACE, indeed, it would have been a contradiction if LINSPEACE were closed under polynomial time reduction, but LINSPEACE is not closed under polynomial time reduction.

Theorem 11 ***CON_x** and **CON** belongs to LINSPEACE, and hence, also to PSPACE.*

It is a suitable exercise for the reader to verify that Theorem 11 holds: First, argue along the lines of the example in Section 5.2 that it is possible to solve **CON_x** in linear space, then, argue that if **CON_x** can be solved linear space, **CON** can also be solved in linear space.

Definition 11 says that a problem is PSPACE-complete if it in addition to be PSPACE-hard, belongs to PSPACE. Thus, the next corollary is a straightforward consequence of Theorem 10 and Theorem 11.

Corollary 3 ***CON_x** is PSPACE-complete.*

What about **CON**? The informal discussion in the beginning of the section concluded that the problem seems harder than **CON_x**, and thus, since we know that the problem is in PSPACE, one should expect the problem to be PSPACE-complete. Well, we have not been able to prove that **CON** is PSPACE-complete, and maybe it is not. Hence, we have an open problem seeming quite interesting from a pure complexity-theoretic point of view.

Open Problem 1 *Determine the complexity of **CON**. Is the problem PSPACE-complete? Does the problem belong to NP or co-NP? Does the problem belong to P?*

7 Conclusions

7.1 A summary

Let us ask ourselves if we have become any wiser as a result of our analysis, and if so, in what respect? Well, what did we want to know? If we were asked by layman, we could say that we wanted to know the answer to the following question: (i) *How*

hard can it be to configure a computing system? In some sense or another, this is the question that motivates our investigations; but there are many sensible ways to approach such an informal question.

We might, for instance, discuss configuration management in light of our past experiences; study the practice of system administrators; study the user-friendliness of software tools for configuring systems[29] and so on. We have chosen, however, to approach the question (i) mathematically, and in order to do so, we had to turn (i) into a mathematical question. The first step in the processes was to introduce the notion of an operator, and we said that to configure a system according to a specific policy amounts to recognising certain properties of operators.

Thus, we transformed (i) into the following question: *(ii) How hard is it to decide if an operator is idempotent, injective, fixed-point convergent etc.?* Now, (ii) is far more suited to analysis mathematically. In order to undertake a complexity-theoretic analysis, we had to turn (ii) into an even more specific question. We had to formalize the operators by introducing the operators expressions, etc.

Following this chain of reasoning to the end, we were able to relate the problem of policy-directed operators to other problems of known complexity and therefore determine a series of answers. The most important of these answers (since this is the one that answers the main question) is that the complexity of **CONx** is PSPACE-complete (note that this is harder than the worst case estimate of a similar but differently formulated problem according to the combinatoric formulation of Sun and Couch, but the additional difficulty goes hand in hand with additional benefits).

What does the result mean? Are we sure that our formulation is the right one, or an interesting one? Ask a stupid question, get a stupid answer. Garbage in, garbage out, etc. There are plenty of expressions to express this well-known adage. We believe that the following is the significance of the result:

- In the absence of any special restrictions, and irrespective of the tools used, the problem of planning, solving and deploying a solution with a deterministic behaviour is at least NP hard, even at time $t = 0$. The presence of an environment might already start to mess with the system after this. This means that system administrators and system designers are forced to employ heuristic methods to solve this matter. Heuristic methods are a plausible route for solution, but the price for using them is an intrinsic uncertainty in the result. This uncertainty is probably no worse than the uncertainties introduced by the uncertain environment after $t = 0$.
- By limiting the form of allowed policies, their corresponding configurations and hence expected behaviours to a set with specific properties, one can find solutions with inherently less complexity that are verifiable. We do not currently understand the limitations imposed by this.

So, are we forced to approximate or limit the complexity of configurations in order to have a reasonable chance of understanding and verifying them? There is a certain practical value to approximation, since (in contrast with the determination of an algorithm with lasting or reusable value) a general policy based solution in a changing environment will require change and adaptation regularly — thus one cannot afford to use algorithms that take too long.

It is actually hard to put oneself in the situation we are proposing. We are almost never in the position of not knowing the answer in the way we suggest here. We have many heuristic rules for fixing problems that are quickly implemented, but it is important to realize that we have no way to verify their correctness, except empirically.

Thus perhaps the real question of interest, for the future, is how is the complexity of a problem reduced by constraining it? Might this lead to provable solutions to configuration problems, or are we stuck with empiricism?

7.2 Approximation or bust

If we want to solve the general system administration problem in terms of operators then we know, either from the arguments here or from the parallel model developed by Sun and Couch, that the problem is at least NP hard. But what if we do not want to solve the general problem? What if we are willing to approximate it, or live with certain restrictions?

It seems intuitive that, if we restrict the problem by imposing additional constraints on the operators, then it might be possible to reduce the inherent complexity. Sun and Couch argue that this is true in restricting to the notion of observable states, for instance; Burgess argued that the fundamental hypothesis can be true by identifying policy as the set of achievable behaviours[6]. Indeed, suppose we take the matrix representation use in the examples earlier: then the search space of these square objects filled with only ones or zeroes is quadratic, and the complexity of solving problems with such operators would fall within the tractable P.

The price paid for such a restriction must be that not all configurations or policies are then implementable. The number of implementable policies would have to be reduced as a result. Just how much they would be reduced is not clear. The answer lies in the grey areas of the fundamental hypothesis, for which we have no detailed answers, but this remains to be determined by future research, at least in idealized systems. Real systems are probably too informal to admit a simple answer.

So the question “What is an acceptable policy?” takes on a new significance after these studies. It could be that the answer to the question is that it is a policy that we can find in a reasonable length of time, rather than one that satisfies every detail of our wishes.

System administrators often speak of “best practices” rather than “optimizations”. This is a disturbing idea for a scientist who immediately wants to know: what is the criterion for best? Perhaps one answer is the closest policy that is achievable within polynomial time by some method? For now, this does not tell us anything new however: even NP hard problems can be verified in polynomial time if one makes the right lucky guess (e.g. if a human administrator or Monte Carlo algorithm performs a short random search and comes up lucky). There is clearly work to be done to clarify these matters.

7.3 Last words

There are many definitions of complexity in science. We have chosen to look at a single one that is related to the difficulty of problem solving. This is a useful measure because it tells us something about how hard it might be to find a ‘guaranteed’ solution to a problem. Alternatives exist: for example, the Kolmogorov complexity, which is, in turn, related to Shannon entropy[30], is a measure of how much variability there is in a pattern of data. It is the length of the shortest computer program that could generate the pattern; however it does not tell us how hard it might be to find that program, so it does not help us with the management of problems.

We have chosen to specialise arguments by talking about bit strings. These bit strings are quite easy to imagine in terms of configuration management problems, but the conclusions are, of course, general, as we illustrated in fig. 5. One can

always build higher structures out of bit strings in polynomial time.

We have used the algorithmic definition of complexity to talk about the complexity of management problems in Network and System Administration. We have found that, in general, solving problems predictably and in a guaranteed fashion is hard (NP or even PSPACE hard). However, if we can live with certain restrictions (if we can formulate our systems within certain reasonable constraints and assumptions) then the problems can be solved in polynomial time. Of course, one is never guaranteed to be able to find such a representation for the task one wishes to accomplish. We are probably doomed to live with the need for approximation.

A pragmatic strategy for management would therefore be to pursue policies in terms of those operations that are easily computed. The question then remains what one might be missing in such a framework. Alas, the margin is too small to answer that question here.

References

- [1] M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, 1979.
- [2] P. Odifreddi. *Classical Recursion Theory, Volumes I and II*. North Holland, 1989, 1999.
- [3] M. Burgess. Configurable immunity model of evolving configuration management. *Science of Computer Programming*, 51:197, 2004.
- [4] A. Couch and Y. Sun. On the algebraic structure of convergence. *LNCS, Proc. 14th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management, Heidelberg, Germany*, pages 28–40, 2003.
- [5] A. Couch and Y. Sun. On observed reproducibility in network configuration management. *Science of Computer Programming*, 53:215–253, 2004.
- [6] M. Burgess. On the theory of system administration. *Science of Computer Programming*, 49:1, 2003.
- [7] M. Burgess. *Analytical Network and System Administration — Managing Human-Computer Systems*. J. Wiley & Sons, Chichester, 2004.
- [8] A. Couch, J. Hart, E.G. Idhaw, and D. Kallas. Seeking closure in an open world: A behavioural agent approach to configuration management. *Proceedings of the Seventeenth Systems Administration Conference (LISA XVII) (USENIX Association: Berkeley, CA)*, page 129, 2003.
- [9] A. Couch and N. Daniels. The maelstrom: Network service debugging via "ineffective procedures". *Proceedings of the Fifteenth Systems Administration Conference (LISA XV) (USENIX Association: Berkeley, CA)*, page 63, 2001.
- [10] B. Hagemark and K. Zadeck. Site: a language and system for configuring many computers as one computer site. *Proceedings of the Workshop on Large Installation Systems Administration III (USENIX Association: Berkeley, CA, 1989)*, page 1, 1989.
- [11] M. Burgess. A site configuration engine. *Computing systems (MIT Press: Cambridge MA)*, 8:309, 1995.

- [12] S. Traugott and J. Huddleston. Bootstrapping an infrastructure. *Proceedings of the Twelfth Systems Administration Conference (LISA XII) (USENIX Association: Berkeley, CA)*, page 181, 1998.
- [13] Webmin project. <http://www.webmin.com>.
- [14] R. Osterlund. Pikt: Problem informant/killer tool. *Proceedings of the Fourteenth Systems Administration Conference (LISA XIV) (USENIX Association: Berkeley, CA)*, page 147, 2000.
- [15] IEEE. Glossary of software engineering terminology, standard 729-1983. Technical report, IEEE, 1983.
- [16] A. Couch and M. Gilfix. It's elementary, dear watson: Applying logic programming to convergent system management processes. *Proceedings of the Thirteenth Systems Administration Conference (LISA XIII) (USENIX Association: Berkeley, CA)*, page 123, 1999.
- [17] M. Sloman. Policy driven management for distributed systems. *Journal of Network and Systems Management*, 2:333, 1994.
- [18] M.S. Sloman and J. Moffet. Policy hierarchies for distributed systems management. *Journal of Network and System Management*, 11(9):1404, 1993.
- [19] M. Burgess. System administration as communication over a noisy channel. *Proceedings of the 3rd international system administration and networking conference (SANE2002)*, page 36, 2002.
- [20] C.E. Shannon and W. Weaver. *The mathematical theory of communication*. University of Illinois Press, Urbana, 1949.
- [21] H. Lewis and C. Papadimitriou. *Elements of the Theory of Computation, Second edition*. Prentice Hall, New York, 1997.
- [22] M. Burgess. Theoretical system administration. *Proceedings of the Fourteenth Systems Administration Conference (LISA XIV) (USENIX Association: Berkeley, CA)*, page 1, 2000.
- [23] G.R. Grimmett and D.R. Stirzaker. *Probability and random processes (3rd edition)*. Oxford scientific publications, Oxford, 2001.
- [24] R. RIZZI. Complexity of context-free grammars with exceptions and the inadequacy of grammars as models for xml and sgml, 2002.
- [25] M. Burgess. Configurable immunity for evolving human-computer systems. *Science of Computer Programming*, 51:197, 2004.
- [26] Seppo Sippu and Eljas Soisalon-Soininen. *Parsing theory. Vol. 1: languages and parsing*. Springer-Verlag New York, Inc., New York, NY, USA, 1988.
- [27] D. Wood. *Theory of Computation*. J. Wiley and Sons, 1987.
- [28] A. Vardy. Algorithmic complexity in coding theory and the minimum distance problem. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing*, pages 92–109, 1997.
- [29] A. Brown, A. Keller, and J. Hellerstein. A model of configuration complexity and its application to a change management system. In *Proceedings of the IXth IFIP/IEEE International Symposium on Integrated Network Management IM'2005*, pages 631–644. IEEE, 2005.

- [30] T.M. Cover and J.A. Thomas. *Elements of Information Theory*. (J.Wiley & Sons., New York), 1991.

System Administration and the Scientific Method

M. Burgess

November 20, 2005

1 Introduction

Networks and computer systems are objects to be observed and understood by a process of scientific inquiry. Even in isolation, computers are too complicated to predict from purely deterministic principles of logic and reason; then, once we place them in a non-deterministic environment of users and networks, the task becomes impossible. The result is that network or system manager is in the position of the natural scientist observing unknown phenomena and learning about behaviour that he or she strives to explain.

Computer Science looks traditionally to logic and mathematics rather than to natural philosophy to describe computers, examining the consistency of assumptions rather than the reality of their implementation. Clearly such studies are valuable, but they do not represent the *practice* of computer systems. Scientists have long since abandoned such fundamental approaches to studying the real world. Trying to describe the behaviour of the human body in terms of atoms and their basic interactions, for instance, is simply not a tractable problem.

Science has learned to eat humble pie and make do with *suitably idealized approximation*. To defend itself from unreasonable criticism, it has then developed a theory for gauging the uncertainty of the conclusions it reaches. In the engineering world, this scientific approach is not only useful but essential. We can distinguish two complementary approaches to understanding the world. Put very simply:

- Empiricism: to observe, model and explain.
- Mathematics: to assume, formulate and deduce.

Science, or natural philosophy, is the discourse between *observation* and *rational approximation*. It is about finding a suitably idealized description of the world by which we might summarize its essence and use the information to good effect. It begins with empiricism, or observation. Mathematics and logic, on the other hand, are precise constructs based on rewriting assumptions according to unambiguous rules. What goes in, comes out in a new and enlightening form.

There are two reasons for modelling: the first is to make sure that one's assumptions actually lead to the outcomes one imagines, through a plausible chain of cause and effect, and also to within an acceptable margin of error. The second is to make predictions about new scenarios that we have never seen, based on past observation and modelling. The world, of course, never obeys simple laws and models exactly: all models are 'wrong', in a sense, but that need not matter as long as we know how wrong they are likely to be.

System administration is about the planning, deployment and maintenance of human-computer systems. The human element of the system here, since it leads to significant uncertainty. Humans make approximation essential.

2 Modelling

We have good reason to model systems: we would like to improve their designs, reliabilities, efficiencies and integrities of purpose. Computers are, above all, technological innovations, i.e. they are tools that are designed to perform a function. We must see them in this role.

This makes them subtly different from natural phenomena, which simply exist for better or for worse. Thus, when applying the methods of science to describe technological innovations, we want to understand and account for how the prejudice of demanding ‘success in fulfilling its purpose’ alters our objectivity and judgements about a device.

For example, we commonly talk about:

- Optimization: the tuning of a device parameter in such a way that it maximizes some derived quantity, such as service level or the number of business transactions per second.
- Policy: the making of essentially ad hoc decisions about what the technology is supposed to do, how it should behave, its goals and the acceptable parameters for its functioning. We cannot make value judgements such as good or bad, right or wrong, without a statement of the rules of the world in which we place our technology. Such things cannot be derived from any deeper knowledge, they must simply be decided and defined.

Unlike the natural world, which has no agenda, the technological world has many, possibly conflicting goals and we must take this into account in modelling. In some respects, policy defines the laws of our technological universe.

The two approaches are distinct in their relationship to modelling, as noted by the philosopher of science David Hume:

- Empiricism: what we observe is real, but we cannot be certain that we have observed is the “truth”. Another observation might disprove any conclusion we might draw from observation. Our observation might even be prone to error, or noise.
- Mathematics and logic: we can prove statements to be true or false, but only within a construct that is, at best, only based on reality. The result is not a truth about the real world, however. It is only a truth about our model.

We can, of course, try to validate the results of a model-construct by experiment, but then we are in the realm of empiricism again, where observation is uncertain. The best we can do is to obtain a bulk of evidence in support of a point of view. A good scientist does not believe that a theory is correct, only successful in describing the uncertain observations he or she makes.

One of the most pragmatic approaches to science has been Karl Popper’s notion of falsification: namely, that it is sufficient to find one counter-example to a theory to invalidate it, but one can never fully validate a model. Thus models remain models, with intrinsic uncertainty, and never ascend to the position of truth.

Many philosophers have pondered these issues[1]. These distinctions are important to the research scientist as well as to the engineer, as one must routinely protect oneself against abuses of science by marketers. Phrases such as “scientifically proven” are clearly nonsense, and yet we read them on a daily basis.

3 Measurements and scales

Whether our goal is to reason about the world, or to measure it, we need scales of measurement. In terms of computer science engineering dimensions can be thought of essentially

as a *type theory* of measurement. All measurements are simply pure numbers, but they are derived from observations made using different devices, over different time scales. For example, both bytes per second and bits per second are data rates, but are they the same? Clearly not.

How old are you? I am five. Five what? Years? Months? Kilogrammes? Metres? Just as it makes no sense to assign a text string to a number, so we must be able to assign like quantities in measurement. If we wish to relate height in metres to age, then we need a theory and a formula for that relationship, which makes sense.

We call the units of measurement *engineering dimensions*, and the square brackets are normally used to denote the dimensions of an object..

$$[\text{data rate}] = \frac{\text{bits}}{\text{seconds}} \quad (1)$$

Dimensions are useful because they help us to identify the correctness and form of relationships. Suppose we suspect that the weight W (kilogrammes) of fibre optic cable is related to its length L (metres). Dimensions tell us that they cannot be equal without a multiplying constant of proportionality:

$$\begin{aligned} W &= \alpha L \\ [W] &= [\alpha][L] \\ \text{kilogrammes} &= \frac{\text{kilogrammes}}{\text{metres}} \times \text{metres} \end{aligned} \quad (2)$$

Thus α must have units kilogrammes per metre. Similarly, if $a = b + c$, then a , b and c must all have the same dimensions. These are elementary observations, but important.

4 Experiments and data collection

To gather information about a system we must observe it and classify our observations into useful categories. Some observations are quantitative, while others are qualitative: Qualitative distinctions present a challenge to deal with: is the horse red or brown? Is a computer PC or a workstation? Where exactly is the borderline between the two? Clearly one must choose classifications with some care to avoid unnecessary ambiguity.

In many cases, one deals with the performance of systems that have measurable data or work rates, using *time series*. A time series is a sequence of measured values at different times, see fig. 1.

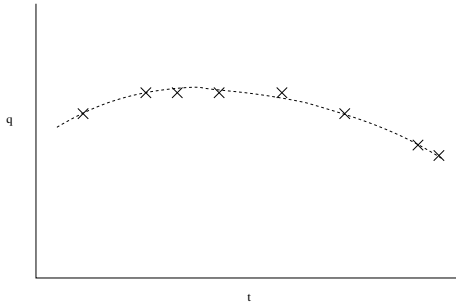


Figure 1: A series of values q marked by crosses are measured at different times t . The dotted line interpolates a proposed smooth curve passing through the points. When is it appropriate to assume such a curve?

A measurement is represented as a cross on this time series, because it is a single event in space and time. The diagram shows a dotted line interpolating the measurements. When and why is it appropriate to assume such a line?

- If we believe the value q is an evolving value that we have only managed to sample at limited certain intervals due to limitations on our resources (music sampling is an example of this).
- If the time scale for changes in q is, say milliseconds, and we measure every hour, then we can clearly see a trend, even if the actual process taking place is not continuous. At the time scale of hours, the individual events seem to form a continuous curve. Relative time scales are highly important.

In these cases, we can assume the existence of a possibly smooth function $q(t)$. If we believe that we are not justified in assuming that q evolves through a number of intermediate values between each observation, then we should not draw a curve through the points, unless we wish to approximate a *trend*. A trend is always a smooth characterization of change, but it is understood to be an indicator rather than a observation about the system.

5 Errors and uncertainties

Scientists speak often about the ‘theory of errors’, since a major source of doubt about results, historically, has been due to human inconsistency. However, even if we could eliminate inconsistency and mistake from the process of measurement, e.g. by using reliable machines to assist us, there are still reasons for doubting the correctness of measurements. A more accurate name is therefore the ‘theory of uncertainties’.

Many uncertainties occur because the act of measurement takes place in an environment over which we have no control. The environment can influence the system we are measuring of the measuring instrument itself.

For example, if we use a computer to measure a precise value at a precise time, the unfortuitous arrival of a disk interrupt could delay the registration of a measurement by some milliseconds and lead to an error, of the same order, in the time. Since one can never be certain that this will not happen, there is an intrinsic uncertainty in the value of the time, due to activities beyond one’s control.

Similarly, the value being measured could be affected by the measurement process. Suppose one is measuring the amount of memory in use: by starting our measurement program, we alter the amount of memory used by the system, and this change might not be exactly predictable, due to differences in paging algorithms.

We identify two types of ‘error’ or ‘uncertainty’:

- Random error: inconsistently fluctuating values that might add or subtract to a measured value, making its value too large or too small at random.
- Systematic error: a constant shift in the value of a measurement due to a consistent pattern of error. For example, forgetting to calibrate the zero mark on a measuring device could make all values measured wrong by the same amount.

With regard to time series, we can identify two kinds of uncertainty:

- Scatter: uncertainty in value (vertical position in fig. 1)
- Jitter: uncertainty in time (horizontal position in fig. 1)

One way to estimate random error is to take multiple measurements. If the value we measure varies on repeated measurement, we should collect sufficient data to be able to determine whether there is any pattern to the variation (see figs. 2 and 3). Data points

that are measured repeatedly are generally plotted as a mean value with an “error bar” that denotes the estimated uncertainty. This makes it easy to gauge the uncertainty by inspection.

There is a number of possibilities that might arise:

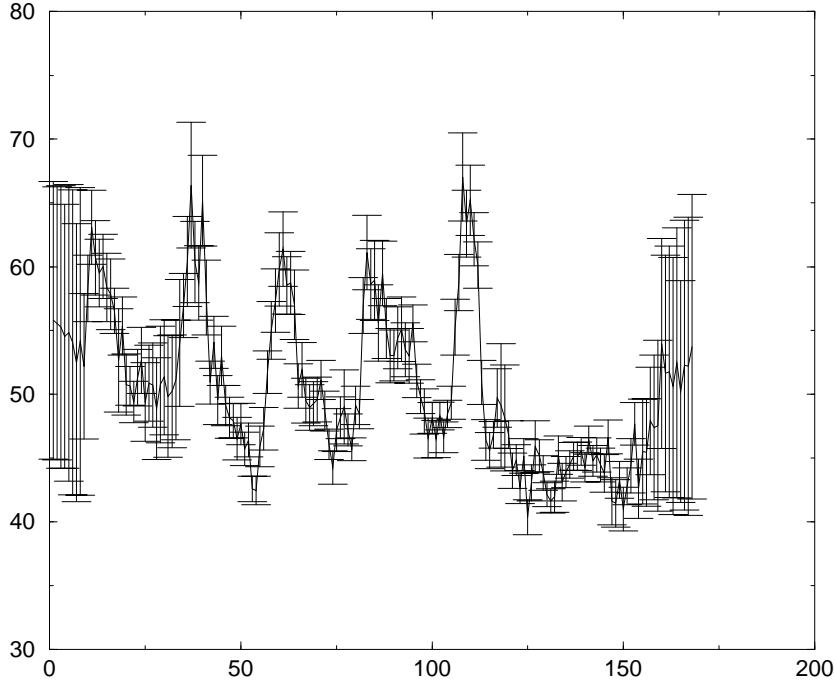


Figure 2: A series of data in which several measurements have been made at each time interval. The horizontal axis measures granular time, and the vertical axis measures the value $q(t)$. The mean value is plotted as a solid line, with an error bar at each measurement showing $\pm\sigma$.

- We always obtain same measured value: the variation or uncertainty in the value is not measurable within the framework of the experiment. Note that this does not necessarily mean there is no uncertainty, because the measuring apparatus itself has a limited resolution. A single value is populated in the histogram $N(q)$.
- We obtain a distribution of values with a mean or expectation value. The expectation value is thus indicative of a ‘true’ value. A preferred value can be seen in the histogram $N(q)$.
- There is no discernable pattern to the values, the value is random without preference. The histogram $N(q)$ is flat, or has no significant peaks.

Several myths abound about the statistics of measurements. It is often assumed that the distribution of measured values forms a Gaussian distribution or approximating histogram [2, 3]. Without qualification, this is an erroneous assumption to make. However, for the case of random errors in independent measurements, the Gaussian model is reasonable and derivable from a mathematical model.

Any distribution of values forms a histogram (or limiting histogram forming a smooth curve) whose axes are q , the actual value measured, and $N(q)$, the number of times the value q occurred in all measurements (see fig. 4). Since we associate meaning with patterns of behaviour, a peak in the histogram signifies a repeated pattern of behaviour in a system.

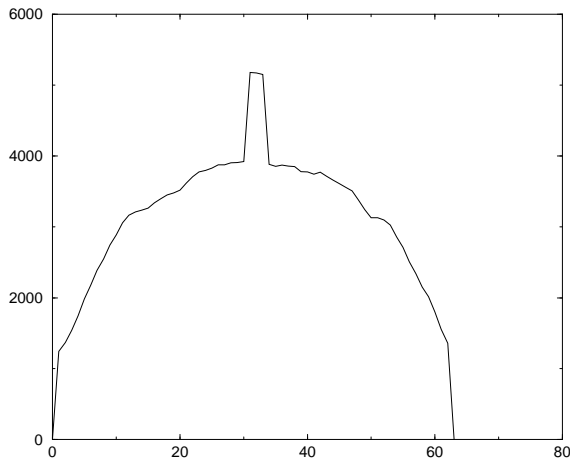


Figure 3: The histogram of the deviation of the actual data points in fig. 2. The horizontal axis shows position about the mean (in the centre), and the vertical axis shows frequency.

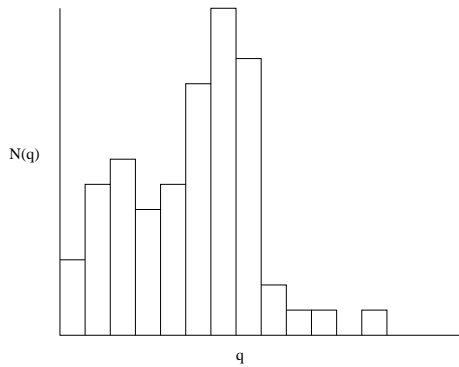


Figure 4: A histogram of values of q shows the frequency with which certain values have been measured. A pattern here signifies important information about whether there is anything deeper going on in the system, or whether the values are truly without meaning.

A single peak indicates the possibility of a ‘true’ value in the measurements. However, we must still explain the reason for the scatter.

- Is it purely due to random environmental interference?
- Does the pattern signify an actual, genuine change in the value measured from the system over time, combined with environmental uncertainty??
- Does the distribution represent different sub-populations of events within the total mass, possibly implying multiple explanations for the results?

Without further information, we cannot tell the reason for scatter. A rational approach to understanding scatter is to formulate a *cause tree* documenting all of the possible causes of variation, and assigning probable cause[1].

Another myth is that the use of a standard deviation assumes a Gaussian distribution. This is false. The standard deviation is merely the second moment, or characteristic, of the distribution (first first moment is the expectation value). It is a characteristic scale for the scatter, whose interpretation must be determined in each case. It does not help us to determine whether the distribution is symmetrical about the mean, for instance, but one

can use it in the spirit that one uses RMS voltages to represent a useful measure of varying voltages from alternating current power sources.

It has a special significance for Gaussian distributions, owing to the fact that the vast majority of the values in the Gaussian form are bounded by two standard deviations on either side of the mean. However, we can still use the mean and standard deviation of values in $(q, N(q))$ to characterize the scatter. In particular, one can use it as an estimate of the uncertainty in the result. to quote a measured value, one therefore quotes the mean, plus/minus a standard deviation.

$$\text{Measured } q \equiv \langle q \rangle \pm \Delta q \equiv \langle q \rangle \pm \sigma_q \quad (3)$$

where $\langle q \rangle = E(q) = \sum_{i=1}^N q_i$ and $\sigma \equiv \sqrt{\sum_{i=1}^N (q_i - \langle q \rangle)^2 / N}$, for sample size $N = \sum_q N(q)$ ¹. We note the limitations of this, when $N(q)$ is highly asymmetrical about the mean.

Some authors use the standard error of the mean σ/\sqrt{n} , in n repeated trials, as the uncertainty. This makes the assumption that there is a single ‘correct’ value to be determined from the measurements, and avoids overestimating the uncertainty in that case. This is a special case, however, and one cannot make such a reduction in general.

Taking account of errors and uncertainties in algebraic expressions is a challenge, but there is a simple approach, which again works best on an approximately Gaussian random uncertainty model. Each individual source of error is quoted as in eqn. (3) If several measured quantities are combined into a function of several independent variables, then each is measured independently, and the uncertainties are combined by a Pythagorean, orthogonal sum. For example, suppose we have a function of several quantities $f(q_1, q_2, \dots, q_n)$, and the functional form of f is known, then we can construct:

$$\Delta f = \sqrt{\left(\frac{\partial f}{\partial q_1} \Delta q_1\right)^2 + \left(\frac{\partial f}{\partial q_2} \Delta q_2\right)^2 + \dots + \left(\frac{\partial f}{\partial q_n} \Delta q_n\right)^2} \quad (4)$$

6 Hypotheses

Understanding data requires, in essence, constructive guesswork. Science is an art as much as it is a discipline: one must have the imagination to think of likely causes in order to be able to test whether the guess is correct or not. Our main aim is to find the probable cause of an observed phenomenon or source of uncertainty. If we can identify the cause, we have a chance of being able to control it, improve upon performance, optimize the result etc.

One way of mapping out probable causes is to use a causality diagram, or cause tree[1, 4] (see fig. 5). This is essentially the same a ‘fault tree’[5, 6] is a way of identifying the possible channels of cause and effect in a system. Each branch can be assigned a certain probability which may be combined with rules logical rules for combination of probabilities into a final estimate of the probability of the outcome (top box). The most interesting aspect of this kind of exercise is in identifying the most likely pathways in the tree, since these are the dominant causes of the effect. In the case of uncertainty estimation, this gives us a clue as to how to reduce the total uncertainty in the answer.

7 Discrete and continuous modelling

The distinction between qualitative and quantitative descriptions of phenomena mirrors a similar dichotomy of representation in modelling systems.

¹Note that it is common practice to use $N - 1$ in the denominator of the standard deviation for a small sample to prevent underestimation by numerical accuracy.

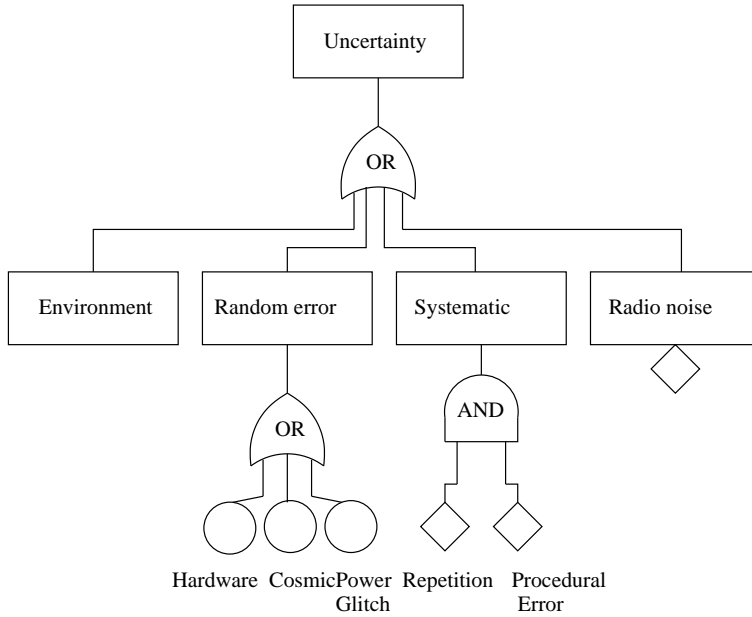


Figure 5: Ordering the hierarchy of possible causes as a cause or fault tree.

Certain problems lend themselves to *discrete* variable modelling, while others require a *continuum description*. Continuum descriptions of phenomena are always approximations, because there is always a limit to the amount of actual detail that can be perceived, e.g. smooth shapes in engineering must eventually make way for the atomic lumpiness of matter; similarly, smooth time-variation in computers must eventually give way to the resolution of the system clock.

- Continuous: infinitely detailed, using calculus of variations to describe change. In a mathematical sense, continuous means smooth or *differentiable*. e.g. Probabilities form a continuum of values in the interval $[0, 1]$.
- Discrete: a countable set of values can be measured. Derivatives do not usually have a straightforward meaning in this case. A discrete variable takes values in a finite set e.g. $q \in \{Windows, Linux, MacOS\}$

Stochastic or event-driven systems are cases where we have discrete arrivals, but the arrivals can occur within a continuum of time. Moreover, the average number of arrivals per second is a pseudo-continuous variable, since an average is a rational number.

How we model and represent change in a system depends on which of these representations we choose.

- Continuous: characterizations can often be formulated as differential equations, with smooth derivatives that develop according to well known rules based on the the rates of change.

$$\frac{dq}{dt} = \lim_{dt \rightarrow 0} \frac{q(t + dt) - q(t)}{dt}. \quad (5)$$

- Discrete: Changes jump between the discrete set of states q_i (see fig. 6). A transition matrix may be used to characterize the allowed transitions as an automaton. One may speak of the conditional probability of making a transition from q_1 to q_2 , in a

stochastic system,

$$P(q_i|q_j) = \begin{pmatrix} P(q_1|q_1) & P(q_2|q_1) & \dots & P(q_n|q_1) \\ P(q_1|q_2) & P(q_2|q_2) & \dots & P(q_n|q_2) \\ \vdots & \dots & \ddots & \vdots \\ P(q_1|q_n) & \dots & \dots & P(q_n|q_n) \end{pmatrix} \quad (6)$$

If this probability is 1 for each possible transition, then the system is said to be deterministic.

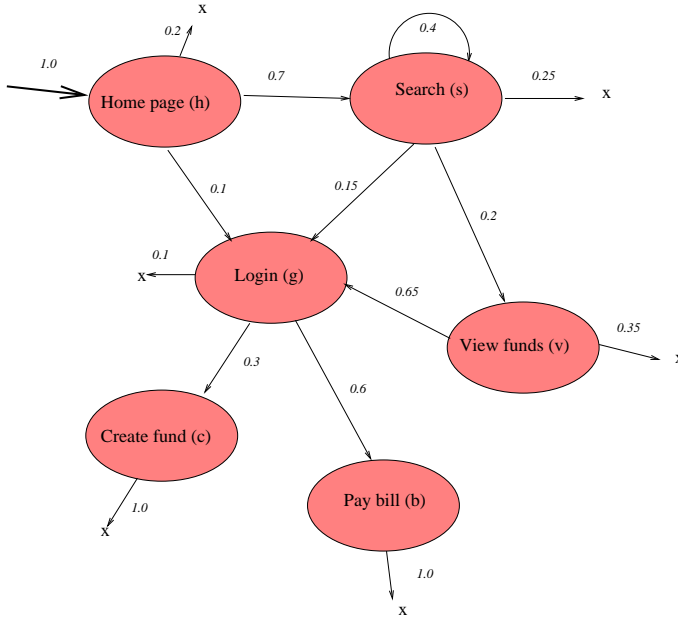


Figure 6: A transition diagram as a graph. The matrix of probabilities can be assembled from the diagram..

8 Modelling responses

In computer systems we are often interested in the response of systems to input. This can be handled both in a continuum and discrete kind of model.

- Continuum: we consider the arrival of a signal current $J(t)$, as a function of time, and measure the response of the system to its arrival. The continuum analogue of the transition matrix is the *response function*, which is formally Green's function of the system. The state $q(t)$ of the system at time t_f is then

$$q(t_f) = \int_{t_i}^{t_f} R(t, t') J(t') dt', \quad (7)$$

given that the signal began arriving at time t_i .

- Discrete: here we consider the arrival of discrete symbols belonging to an alphabet $\Sigma = \{A, B, \dots\}$. These might represent user commands or protocol transactions,

etc. The system state q is now determined by a transition matrix, and by the particular symbol that arrives. The result is a set of *labelled transitions*:

$$q_i \xrightarrow{A} q_f. \quad (8)$$

This is to be interpreted as: if the system is in state q_i and receives symbol A , from the alphabet, then a transition is made to state q_f .

We would like to have ways of identifying patterns of input in order to associate meanings to them.

In a continuum approximation, we can simplify the details of individual events into a continuum of behaviour and attempt to *fit* a smooth function. For example, waves impinging on a receiver might properly consist of the arrival of individual photons, but we normally represent the average behaviour as a smooth function such as

$$q(x, t) = \sin(kx - \omega t) \quad (9)$$

This kind of approximation is very useful, because we can write down the entire behaviour in a simple closed form that can be manipulated according to known rules of algebra. As a mathematical model, we do not normally model the uncertainty of observations, but treat this closed form as a theoretical ‘truth’.

In the case of a discrete alphabetic stream of symbols, the *theory of languages* provides a classification of patterns. The Chomsky hierarchy classifies the levels of sophistication in a pattern, called a grammar, by associating each level of pattern with an automaton that can decipher it. More often than not, a complete specification of a grammar is complicated and the most basic pattern structure, a *regular expression* is used to classify patterns of text.

The Chomsky hierarchy is important, for example, in configuration management, where one would like to edit the contents of a file or database. A file editor that understands structures like recursively bracketed blocks of symbols is clearly able to provide a much more sophisticated understanding than a file editor which is only able to match simple strings of characters. Regular expressions (and regular languages) are helpful in summarizing the essence of structured interactions, like protocols. An extremely simple representation of a TCP stream might therefore be written,

$$"[SYN][ACK] * . * [FIN]" \quad (10)$$

where the ‘.’ (dot) character stands for any symbol in the alphabet.

9 Configuration management and maintenance

As an example of these modelling forms, let us consider the highly important example of computer configuration and maintenance. Placing a system in a certain state and trying to keep it there is an important part of management policy. One must take into account the stochastic environment, which tends to push the system into neighbouring states from which one must recover.

To model this, we use our two approaches:

- Discrete: each change is an operation that transforms a state q into a new state q' , by some transition matrix.
- Continuous: look at the long-term trends $q(t)$ and find the rates at which changes happen. What does that mean about the need for management?

Changes are introduced

- By us (intentional change).

- By accident (random error).

Later we shall relate changes to information theory in order to discuss reliability.

As always, we use continuous models to discuss rates of change, and discrete models to discuss the microscopic details of individual changes.

9.1 Discrete configuration

Configuration management is the discrete form of regulation theory. Policy can be thought of as a specification of the state in which we wish the system to reside. Since changes and errors can be introduced by users and administrators at any time, one must ensure that this is the case by performing regular maintenance.

Configuration management includes both hardware and software configuration in

- The set up of the system.
- The maintenance of the system.

Suppose we represent file or other system objects by a vector. We can think of this as a vector of bits[7]:

$$\vec{q} = \begin{pmatrix} q_1 \\ q_2 \\ q_3 \end{pmatrix} \quad (11)$$

where e.g. q_1 is a file, q_2 is a process, q_3 is an access permission etc. One can make any abstraction that is convenient.

To make a change $q \rightarrow q'$, we need a transition *matrix* or *operator*. This is a convenient representation of the problem. The operation of repairing a broken state is written:

$$\vec{q}_{\text{fixed}} = \hat{O}_{\text{repair}} \vec{q}_{\text{broken}} \quad (12)$$

i.e. in explicit form one can write larger matrix generalizations of

$$\begin{pmatrix} 1 \\ q'_1 \\ q'_2 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & ? & ? \\ 0 & ? & ? \end{pmatrix} \begin{pmatrix} 1 \\ q_1 \\ q_2 \end{pmatrix} \quad (13)$$

going up to q_n , for arbitrary n . Each change or repair is an operation mediated by operator \hat{O} . We include a 1 as the top-most element of the vector to enable us to set the boolean states to 1 by copying this value.

Suppose we start with a system ‘out of the box’, in state q_0 . We decide what we want and then set this up by performing a string of operations on it, in order to obtain a policy state.

$$\vec{q}_{\text{policy}} = \hat{O}_n \dots \hat{O}_3 \hat{O}_2 \hat{O}_1 \vec{q}_0. \quad (14)$$

e.g. \hat{O}_1 is edit /etc/passwd, \hat{O}_2 is ‘install software’, etc.. This notation allows us to discuss the algebra of changes.

One must now be careful. Do the operators commute? i.e. Is order important in the application of the changes? If two operations commute

$$[\hat{O}_1, \hat{O}_2] \equiv \hat{O}_1 \hat{O}_2 - \hat{O}_2 \hat{O}_1 = 0 \quad (15)$$

then order does not matter. If the result is non-zero, we might get the wrong answer unless the order is guaranteed.

If the order of a collection of operators matters, then we must know what state we start out from, in order to know what state we will end up in. But if the order does not matter, we have the possibility of arriving at the same end result no matter how we apply the changes.

For example, the configuration management software cfengine works on a philosophy of trying to provide operators that commute[8, 7], so that one can repair systems in real time, with the minimum of interference.

The software IsConf takes the approach that, when an error is detected, one should reset the system back to q_0 and reproduce the exact order of operations[9]. These two approaches are not as different as they sound.

9.2 Discrete convergence

What additional properties, or constraints, should we impose on the maintenance operations \hat{O}_i in order to make maintenance predictable and reproducible? Stability is clearly a powerful concept, related to maintenance. The need to maintain stability in the face of random change means that one must place additional restrictions on the operators.

The stochastic nature of environment can be modelled as a series of random operators arriving and altering the desired state to a new state. Maintenance should therefore be considered as ‘undoing’ all of these operations relative to the current state. Configuration maintenance can then be automated by simplifying continuously applying operations \hat{C} repeatedly, in any order.

Undoing operations caused by environmental change might make one think of looking for the inverse operators that reverse the ordering of changes. However, finding inverse operators for the changes is a difficult problem, and we need not find an inverse in the normal sense.

An alternative is to create an operator that takes us from any state to our policy state, and then no further i.e. if we apply operator to a non-policy state, we get the policy state. If we apply it to the policy state, nothing happens.

$$\begin{aligned}\hat{C}\vec{q}_{\text{broken}} &= \vec{q}_{\text{policy}} \\ \hat{C}\vec{q}_{\text{policy}} &= \vec{q}_{\text{policy}}\end{aligned}\tag{16}$$

This property is called convergence[10], and has the effect of making the policy state *absorbing*, with respect to the operation. It is a generalization of the concept of idempotence for operators[11, 12] which includes a ‘memory’ of the preferred state.

An example of a convergent edit operation would be e.g. \hat{C} is “add mark to passwd file if user mark does not already exist”. What is new is that the operators need to “observe” the state of the system before acting, so that they are aware of their location in the set of possible states.

9.3 Continuous maintenance

Continuous models are approximations over long time-scales or large numbers of operations.

They allow one to make general statements about the rate of damage versus the rate of repair. How quickly do we have to act to maintain trends in behaviour, i.e. policy?

Formally, one obtains a continuum model from a discrete model using a *local averaging procedure* to find a continuum picture. This only works for numerical variables, so it is natural to consider the stochastic point processes which lead to change and their associated counting processes. Alternatively, one can consider an enumeration of the states whose which can be plotted as a time-series (as in fig. 2). In other words, we could the numbers of primitive operators or changes in order to gain insight into the trends.

The continuum model is thus found by dividing time up into regions and using the average value from each region as the smooth trend. Interpolating a smooth curve through these gives one the approximate picture seen when zooming out to long times

A change in each region of the system can be written relative to the local mean value $\langle q \rangle_t$:

$$q(t) = \langle q(t) \rangle + \delta q(t) \quad (17)$$

i.e. signal = trend + fluctuation.

9.4 The maintenance theorem

Based on a fluctuation model of the stochastic state of a system, one can consider the rate at which repairs must be implemented in order to maintain stability over a certain timescale. The theorem says that we should associate policy with $\langle q \rangle_t$, i.e. the average state. We should allow some ‘slack’ into the policy because we cannot define the precise state realistically. So we define policy as the acceptable trends in system development, which means we allow a slack of the order δq .

Now, to ensure that the system does not deviate by more than δq , we need to repair the change within a time Δt , where:

$$\delta q \simeq \frac{dq}{dt} \Delta t \ll q \quad (18)$$

i.e. when the system is dominated by a smooth average development. For example, security policy might be adapting slowly over weeks (small dq/dt), but a rapid change δq should be repaired quickly to maintain the integrity of that policy.

The continuous modelling a system state is important in all cases in which large numbers of transitions are involved. This includes anomaly detection and network monitoring.

10 The concept of a system

The concept of a system is common to a wide range of activities and scenarios, from machines to human enterprises. In general terms, a system can be thought of organized effort to fulfill a role. As a tool, information technology is integrated into other systems in almost every sphere of application, from hospitals to businesses.

In terms of modelling, we need some way of formalizing the activities in a system, measuring and parameterizing the results. Once again, we are led to consider the two approaches to modelling: the discrete and the continuous. The discrete approach excels at detailed interactions, while the continuum approach deals best with trends and rates of change over longer time-scales.

What constitutes a system? A system is a balance between *freedoms* and *constraints*. A freedom is a potential for change within the system. Freedom to change is usually represented by a parameter which can be varied over a range of values. A constraint is a limitation on the possible changes which can occur on variables or parameters in a system. This often takes the form of a rule, or parameter inequality.

Symmetries are descriptions of what can be changed in a system without affecting the system’s function. Determining what is *not* important to the functioning of a system is a way of identifying degrees of freedom that could be manipulated for strategic advantage.

If we wish to describe the behaviour of a system from an analytical viewpoint, we need to be able to write down a number of variables which capture its behaviour. Ideally, this characterization would be numerical since quantitative descriptions are more reliable than qualitative ones, though this might not always be feasible.

It is reasonable to distinguish *static* or *dynamic* systems. A system that is static undergoes no measurable change over the timescales of interest, whereas a dynamical system is

undergoing noticeable change for at least part of the time. This is not to say that a static system is uninteresting. System architectures are static, but transactional processes and resource usage are dynamical.

For instance, a library or database archive is a static organization of data (a data structure). Data models represent aspects of a systematic organization that are not subject to frequent change. A number of such models has been developed for network and system management: for example, DEN-ng[13, 14], eTOM[15].

Principles such as database normalization[16] are used to rationalize and optimize discrete structures.

An *algorithm*, or protocol, is a discrete encapsulation of a *process* in terms of operational steps. Flow models and networks can also be used to apply continuum approximations to Modelling schemes for processes include Petri nets[17, 18] and UML[19, 20] to name only a couple[21].

Inventory resource models[22] and schedules are also continuum approximations to discrete processes, which consider methods of optimization.

A final categorization of dynamical systems fall is to consider their descriptive completeness, and hence the uncertainty involved in their modelling. One speaks of *open systems* (partial systems) and *closed systems* (complete, independent systems).

An open system is essentially a *sub-system* of a larger system. An open system received input from its environment and generates output, i.e. it communicates with its environment. A system is said to be open because input changes the state of the system's internal variables and output changes the state of the environment. Since one cannot normally predict the exact behaviour of what goes on outside of a black box (it might itself depend on many complicated variables), any study of an open system tends to be incomplete. An open system can be internally *deterministic*, in terms of its operation. meaning that it follows strict rules and algorithms, but its behaviour is not necessarily determined, since the environment is an unknown.

A closed system is an idealization: a system which is complete, in the sense of being isolated from its environment. A closed system receives no input and normally produces no output. Computer systems can only be approximately closed for short periods of time. The essential point is that a closed system is neither affected by, nor affects its environment. In thermodynamics, a closed system always tends to a steady state. Over short periods, under controlled conditions, this might be a useful concept in analyzing computer sub-systems, but only as an idealization. In order to speak of a closed system, we have to know the behaviour of all the variables which characterize the system. A closed system is said to be completely *determined*

The principal difference between these is how the principal of causality can be applied. Since most systems are best approximated by open systems, one must turn to stochastic models in which causality is not always directly traceable. Statistical methods and uncertainties must then be used.

11 Diagrammatic modelling

Diagrammatic methods of modelling are manifold[1], from entity relation diagrams, through flow diagrams and graphs. Diagrams are themselves systems with the freedom of space, location, colour, shape, direction. and constraints on size, location, topology, etc. Graphs are of special interest in the modelling of networks of components.

The use of graphs is a powerful way of analyzing relationships between discrete objects (see the chapter by Canright and Engø-Monsen in this volume). One may represent both directed and un-directed relationships, e.g.

- Directed $A \rightarrow B$: dominates, depends on, promises to, etc.
- $A \leftrightarrow B$: is associated with, is correlated with, is adjacent to.

A graph is formally a pair $\Gamma = \langle N, L \rangle$, where N is a set of nodes and L is a set of directed links. The degree of a node k is the number of links it has. For directed graphs one separates in-degree and out-degree, according to the direction of the arrows. Graphs are conveniently represented by their adjacency matrices A .

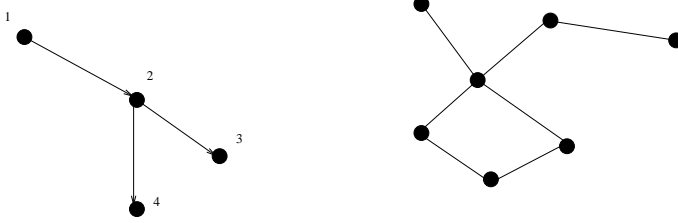


Figure 7: Directed and undirected graphs.

For example, the first graph in fig. 7 is represented by an adjacency matrix of the form:

$$A = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \quad (19)$$

The set of nodes labels the rows and columns, and the links are represented by 1s in the matrix.

11.1 Connectivity

The concept of connectivity is useful in networks of communicating hosts[1]. The connectivity in a graph may be defined as a fraction of the total number of possible directed connections. Let \vec{h} be a vector of n host-nodes in a network. Let \vec{h} be a vector of node availabilities, i.e. the i th component h_i , $i = 1 \dots n$, is 1 if the host-node i is fully available, and zero if it is completely unavailable. Then, if A is the adjacency matrix, the scalar connectivity χ may be defined as

$$\chi = \frac{1}{n(n-1)} \vec{h}^T A \vec{h}. \quad (20)$$

χ has a maximum value of 1 when all nodes are connected bi-directionally and are fully available. It has a value of zero if the nodes are either unavailable or disconnected. This is a useful scalar measure of a network that is easily computed.

11.2 Continuum functions on graphs

A vector of values \vec{h} can be thought of as defining a function, of graph nodes, i.e. a surface (see fig. 8).

Such a vector can be used to characterize local properties of the graph in a neighbourhood of a node. One measure of particular interest is the eigenvector centrality. The relative centrality of a node i is given by the i th component of the principal eigenvector belonging to the adjacency matrix, i.e. let ρ be the largest positive eigenvector of the adjacency matrix A , then

$$A\vec{\rho} = \rho\vec{\rho}, \quad (21)$$

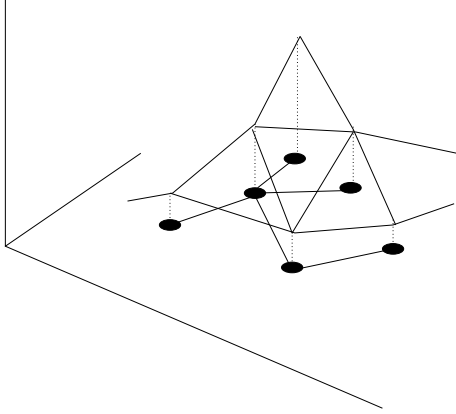


Figure 8: A vector represents a ‘height’ function based on the discrete points of a graph. This is easily visualized as a rough surface.

where I_i is the centrality of the i th node.

The value of I_i represents a fixed-point importance ranking for the graph, which is based on topology alone. It is therefore a complementary *vector* measure to the scalar connectivity which measures how well connected a node is. The most well-connected nodes are those which are bottlenecks for communication, as which are the most vulnerable nodes to attack by an agent wishing to destroy the network[23].

11.3 Probabilistic networks

We do not always have full information about the topology of a graph. In that case, one can collect statistics about the degree distributions and make a statistical estimate of the probability for being able to travel to any node in the graph from any other[24, 1]. Consider a graph of n nodes with degree distribution $p(k)$, where $\sum_k p(k) = 1$. The condition for ‘percolation’ through the network is:

$$\langle k \rangle^2 + \sum_k k(k-2)p(k) > \log(n). \quad (22)$$

This can be understood as follows. If a graph contains a giant component, it is of order n and the size of the next largest component is typically $O(\log n)$; thus, according to the theory of random graphs the margin for error in estimating a giant component is of order $\pm \log n$. In the criterion above, the criterion for a cluster that is much greater than unity is that the right hand side is greater than zero. To this we now add the magnitude of the uncertainty in order to reduce the likelihood of an incorrect conclusion.

The utility of this relation lies in being able to gauge the security or penetrability of a network based entirely on a knowledge of average numbers of connections. Although it is not a precise way of modelling, it gives a simple indicator for large and difficult networks.

11.4 Fault trees

Fault trees, such as that in fig. 5 are a useful and quantitative way of analyzing networks. They enable one to combine denumerable causes of failure quantitatively, within the framework of probability theory. One uses logic gate notation to indicate whether multiple branches of the tree are needed simultaneously (AND) or only independently (OR); these individual probability estimates can then be combined as follows. Let p be the probability

of a failure in a component.

$$\begin{aligned}
 p_{\text{AND}} &= \prod_{i=1}^N p_i \\
 p_{\text{OR}} &= 1 - \prod_{i=1}^N (1 - p_i).
 \end{aligned} \tag{23}$$

Fault tree analysis is beyond the scope of this review, however see the excellent book by Rausand and Høyland[4], and the chapter in this volume by Reitan for more details.

12 Discrete streams: information

The theory of languages provides a classification of the discrete models for classifying strings of discrete events (symbols)[25]. This theory, while interesting, has few direct engineering applications. A continuum approximation to this theory was developed by Shannon in the 1940s however, for specific application in the stochastic world of telecommunications[26].

This is somewhat analogous to the statistical theory of graphs in the previous section. Rather than considering the detailed microscopics of transmissions, which would pose an intractable problem in most cases, one can attempt to make the best of things by formulating a theory based on statistics of symbol distributions.

The theory of communication, now known as *information theory*, is a discussion about symbols in a stochastic framework: it considers how symbols can be misunderstood, or lost in transit, it considers errors of communication and their correction, the limits of symbolic compression, the smallest amount of information required to exactly reproduce a state of configuration, and so on.

There are several applications of information theory in the field of network and system administration: clearly the transmission of data by network is an obvious one; however, even at a high level, the most systems can be viewed as a propagation of state from an input to an output, and therefore information theory should also be able to say something about the integrity of systems in general.

The essence of information theory lies in the discrete time-series in fig. 9. A message or system process can be regarded as a stream of discrete symbols (e.g. operators which are causing change, or data packets of different kinds etc.) which are arriving at some kind of detector. The origin of the symbols could be from the system itself, from an external program or even a remote network source.

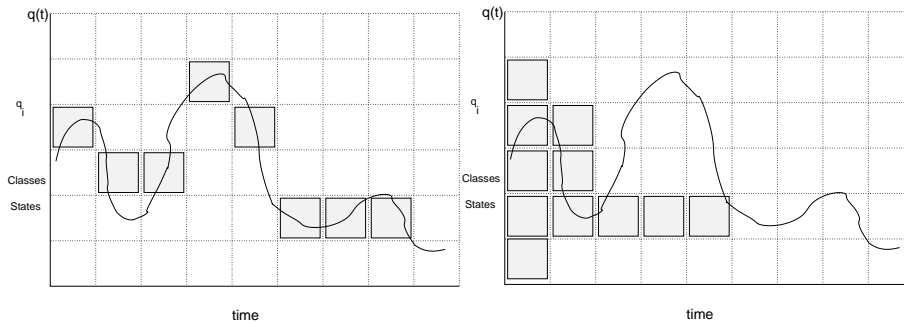


Figure 9: The statistical properties of a time series of discrete symbols can be collected and used to define a theory of information.

There are several aims in this model

- To characterize the average properties of the data stream. This allows one to perform what is essentially inventory control on the symbols.
- To understand what happens when we expose the symbols to an environment which can introduce errors into the data, by randomly altering symbols.

One is thus interested in the extent to which the symbols which were transmitted by the sender (e.g. the configuration state that was asserted by a manager) actually arrives at the system intact (actually remains valid as time passes) as one transmits the data over a channel (as the system is propagated into the future in the presence of noise).

The analogy in the preceding paragraph is the essence of average state and its maintenance, i.e. it is the essence of a system's *policy state*. It links the idea of a stochastic arrival process with that of a stream of symbols of different *types*.

Consider the arrival of different symbols q at different times. The arrivals q fall into a number of classes or types, labelled by $i = 1, \dots, C$. If one collects data in the histogram in fig. 9 for long enough, one can turn the frequency counts for each symbol type i into a probability p_i for the occurrence of that symbol type over time, such that $\sum_{i=1}^C p_i = 1$.

Based on this, it is possible to compute a scalar measure of the information content of a string of symbols, called the Shannon entropy:

$$H = - \sum_{i=1}^C p_i \log_2 p_i, \quad (24)$$

where i runs over the alphabet of symbol types. It is shown in ref. [1] that, in the limit of long messages, H is equal to the average number of bits per symbol that is needed to uniquely reproduce the message, i.e. to distinguish it from another messages based on the same alphabet.

The entropy does not remember the actual message, it provides only a statistical characterization of the amount of bulk information per symbol of the original message.

The entropy has a minimum of zero value when a message consists of a continuous stream of the same symbol, and a maximum value of $\log_2 C$ when all of the symbols in the alphabet are used with equal frequency.

Interestingly, the entropy provides a limit on the extent to which a message can be compressed without altering its content.

Entropy can also be used in an analytical technique of optimization. By maximizing the entropy of a distribution of symbols, one finds the 'most random' or flattest or most widely distributed configuration of possible types, subject to additional constraints. For example, the normal distribution can be understood as the flattest distribution of points with a definite mean value and fixed root-mean square uncertainty (variance). Maximize the Lagrangian, with multipliers α and β enforcing the normalization of probability and the constant variance:

$$L = - \sum_{i=1}^C p_i \ln p_i - \alpha \left(\sum_{i=1}^C p_i - 1 \right) - \beta \left(\sum_{i=1}^C p_i (q_i - \bar{q})^2 / \sigma^2 - 1 \right) \quad (25)$$

$$\frac{\partial L}{\partial p_i} = \frac{\partial L}{\partial \alpha} = \frac{\partial L}{\partial \beta} = 0 \quad (26)$$

$$p_i \propto \exp(-\beta(q_i - \bar{q})^2 / \sigma^2) \quad (27)$$

Consider the transmission of a message from a sender to a receiver. We can call the message *policy*. Can we measure the integrity of the message as it is propagated through the system, over a noisy channel[27]?

Suppose one knows the original message, what is probability of it still being the same after some time along its journey?

We can compare the information (i.e. the entropy) at the start and end of the journey. The conditional entropy $H(O|I)$ is the average information at the output, given that the output is causally connected with the input, i.e. the two are related by more than just coincidence. Similarly $H(I|O)$ is the converse. The overlap between these two $H(I; O)$ (see fig. 10) is the amount of information that is common to both the input and the output, by more than mere coincidence[28]. Mutual information is therefore a measure of the integrity of one's system.

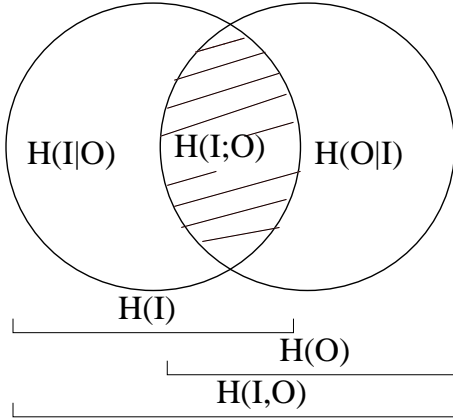


Figure 10: The mutual information related to the other entropies. Here the quantities are shown schematically as measures on an abstract space of all possible messages.

A well known and important formula that results from the informational entropy is the Shannon formula for transmission of a *bandwidth limited* continuous channel, with Gaussian noise. In this case, the representation in figure 9 can be understood as follows. The total height of the graph represents a bandwidth B in Hertz, which could be infinitely divided (and therefore carry infinite information) if it weren't for the fact that noise N (Watts) limits the resolution of the signal S (Watts). The Gaussian spread of noise makes the resolution finite (think of these as the squares or graduations on the diagram): the greater the noise, the larger the squares — i.e. the fewer that will fit into the bandwidth. Thus the question is how many squares can one fit into the bandwidth: this tells us the capacity C of the channel in bits per second. The answer[1, 28] is the classic formula

$$C = B \log_2 \left(1 + \frac{S}{N} \right). \quad (28)$$

The connection between system administration and information theory is not just fortuitous, it is important for several reasons. It relates the matter of system management to a well-known body of knowledge about communication, and a model that is established. It provides a quantitative way of addressing the issue of system integrity using error correction. It allows us to see policy as the transmission of *intent* over time and in the presence of an unpredictable environment, as a system talking to itself over a noisy channel.

13 Continuous streams: queues

Service provision is usually modelled as a relationship between a client and a server (or service provider). A stream of requests arrives randomly at the input of a system, at a rate

of λ transactions per second, and they are added to a queue, whereupon they are serviced at a rate μ transactions per second by a processor.

Generalizations of this model include the possibility of multiple (parallel) queues and multiple servers. In each case one considers a single incoming stream of transaction requests; one then studies how to cope with this stream of work.

As transaction demand is a random process, queues are classified according to the type of arrival process for requests, the type of completion process and the number of servers. In the simplest form, Kendall notation of the form $A/S/n$ is used to classify different queueing models.

- A : the arrival process distribution, e.g. Poisson arrival times, deterministic arrivals, or general spectrum arrivals.
- S : the service completion distribution for job processing, e.g. Poisson renewals, etc.
- n : the number of servers processing the incoming queue.

13.1 The Poisson assumption

The distribution of inter-arrival times for both A and S is normally considered to be a Poisson distribution in discrete time. One writes this $M/M/n$. This assumption is made in order to simplify the analysis. M stands for ‘memoryless’ because the Poisson distribution, taken over non-overlapping time intervals is a Markov process whose behaviour, at any discrete time t , is quite independent of what has happened in the past, i.e. it has no memory of the past. This provides a huge simplification of the analysis.

Another reason for the Poisson assumption is that, in the limit of large numbers of independent arrivals, one would expect the limiting distribution to have a Poisson form. The argument, on very general grounds is based on the fact that a Poisson distribution is the limiting form of any Bernoulli process in which an event either occurs or does not occur at time t . Suppose the probability of obtaining a result is fixed and is equal to p on each independent observation, then the probability of obtaining q positive arrivals in n observations is

$$P(q) = {}^nC_q p^q (1-p)^{n-q}, \quad (29)$$

where ${}^nC_r = n!/(n-r)!r!$ are the binomial coefficients. This is a binomial distribution whose mean value is $\bar{q} = np$. Now, suppose the probability of observing an arrival is scarce i.e. $p \rightarrow 0$, but we consider the limit of long times or many observations $n \rightarrow \infty$, taking the limit in such a way as to make $np \rightarrow \lambda$, where λ is a constant. Then, noting the limits,

$$\begin{aligned} {}^nC_r &\leq \frac{n^r}{r!} \\ {}^nC_r &\geq \frac{n^r}{r^r} \\ \lim_{n \rightarrow \infty} (1-x)^n &\rightarrow e^{-x} \quad \text{where } (x < 1). \end{aligned} \quad (30)$$

one observes that

$$P(q = k) \leq \frac{n^k}{k!} p^k (1-p)^{n-k} \rightarrow \frac{(np)^k}{k!} (1-p)^{n-k} \rightarrow \frac{\lambda^k}{k!} e^{-\lambda}, \quad (31)$$

which is the Poisson distribution, for which one verifies that $\sum_{k=0}^{\infty} P(q = k) = 1$.

This widely held belief is somewhat controversial, however, as measurements of network traffic have shown evidence of considerable ‘burstiness’, or long-tailed behaviour [29, 30, 31, 32]. Other work indicates that these contradictory measurements would in fact settle into a Poisson distribution if only enough measurements were taken [33, 34]. However,

it is estimated that something of the order of 10^{10} transactions might be needed to see this limiting form emerge.

Regardless of this controversy, the Poisson model survives in queueing theory for its overriding simplicity.

13.2 $M/M/s$ queues

The basic ideas about queueing can be surmised from the simplest model of a single server, memoryless queue: $M/M/1$. Then, let n be the number of unprocessed requests in the queue at time t , and suppose that requests for transactions arrived at a rate of λ per second, and can be processed at a rate of μ per second.

We can treat this simple case as a continuum flow approximation, using expectation values.

Consider any time t : the system makes a transition from a state of queue length $n - 1$ to n at a continuous rate λ . Similarly, when it is in state n , it makes transitions to a state of length n at a rate μ . The expectation of λ arrivals per second, when the queue length is in a state $n - 1$ (for any n) must therefore lead to the expectation of having μ completions when the state is n , in order to balance the queue, so we write, on balance $\lambda p_{n-1} = \mu p_n$ or

$$p_n = \rho p_{n-1}, \quad (32)$$

where $\rho = \lambda/\mu < 1$ is called the traffic intensity[1, 35]. This is a recurrence relation that can be solved for the entire distribution p_n , for all n . One finds that

$$p_n = (1 - \rho)\rho^n, \quad (33)$$

and hence, the expected length of the queue is

$$\langle n \rangle = \sum_{n=0}^{\infty} p_n n = \frac{\rho}{1 - \rho}. \quad (34)$$

Clearly as the traffic intensity ρ approaches unity, or $\lambda \rightarrow \mu$, the queue length becomes infinite, as the server loses the ability to cope.

This situation improves somewhat for s servers ($M/M/s$), where one finds a much more complicated expression. In simplified form one has

$$\langle n \rangle = s\rho + P(n \geq s) \frac{\rho}{1 - \rho} \quad (35)$$

where the probability that the queue length exceeds the number of servers $P(n \geq s)$ is of order ρ^2 for small load ρ , which naturally leads to smaller queues.

It is possible to show that a single queue with s servers is at least as efficient as s separate queues with their own server. This satisfies the intuition that a single queue can be kept moving by any spare capacity in any of its s servers, whereas an empty queue that is separated from the rest will simply be wasted capacity, while the others struggle with the load.

13.3 Flow laws in the continuum approximation

Dimensional analysis provides us with some simple continuum relationships for the ‘work-flows’. Consider the following definitions:

$$\text{Arrival rate } \lambda = \frac{\text{No. of arrivals}}{\text{Time}} = \frac{A}{T} \quad (36)$$

$$\text{Throughput } \mu = \frac{\text{No. of completions}}{\text{Time}} = \frac{C}{T} \quad (37)$$

$$\text{Utilization } U = \frac{\text{Busy time}}{\text{Total time}} = \frac{B}{T} \quad (38)$$

$$\text{Mean service time } S = \frac{\text{Busy Time}}{\text{No. of completions}} = \frac{B}{C}. \quad (39)$$

The utilization U tells us the mean level at which resources are being scheduled in the system. The ‘utilization law’ notes simply that:

$$U = \frac{B}{T} = \frac{C}{T} \times \frac{B}{C} \quad (40)$$

or

$$U = \mu S. \quad (41)$$

So utilization is proportional to the rate at which jobs are completed and the mean time to complete a job. Thus it can be interpreted as the probability that there is at least one job in the system.

Suppose a web server receives hits at a mean rate of 1.25 hits per second, and the server takes an average of 2 milliseconds to reply. The law tells us that the utilization of the server is

$$U = 1.25 \times 0.002 = 0.0025 = 0.25\%. \quad (42)$$

This indicates to us that the system could probably work at four hundred times this rate before saturation occurs, since $400 \times 0.25 = 100\%$. This conclusion is highly simplistic, but that is the price for having a simple formula!

Another dimensional truism is known as Little’s law of queue size. It says that the mean number of jobs in a queue $\langle n \rangle$, is equal to the product of the mean arrival rate λ (jobs per second) and the mean response time R (seconds) incurred by the queue:

$$\langle n \rangle = \lambda R. \quad (43)$$

Note that this equation has the generic form $V = IR$, similar to Ohm’s law in electricity. This analogy is a direct consequence of a simple balanced flow model.

In the M/M/1 queue, it is useful to characterize the expected response time of the service centre. In other words, what is the likely time a client will have to wait in order to have a task completed? From Little’s law, we know that the average number of tasks in a queue is the product of the average response time and the average arrival rate, so

$$\begin{aligned} R = \frac{Q_n}{\lambda} &= \frac{\langle n \rangle}{\lambda} \\ &= \frac{1}{\mu(1 - \rho)} \\ &= \frac{1}{(\mu - \lambda)}. \end{aligned} \quad (44)$$

Notice that this is finite as long as $\lambda \ll \mu$, but as $\lambda \rightarrow \mu$, the response time becomes unbounded.

Load balancing over queues is a topic for more advanced queueing theory. Weighted fair queueing and algorithms like round-robin etc., can be used to spread the load of an incoming queue amongst multiple servers, to best exploit their availability. Readers are referred to refs. [1, 35, 36, 37] for more discussion on this.

14 Workflow and scalability

What architecture should one employ in network and system management? An obvious answer is to centralize control in one convenient place. However, this often results in a

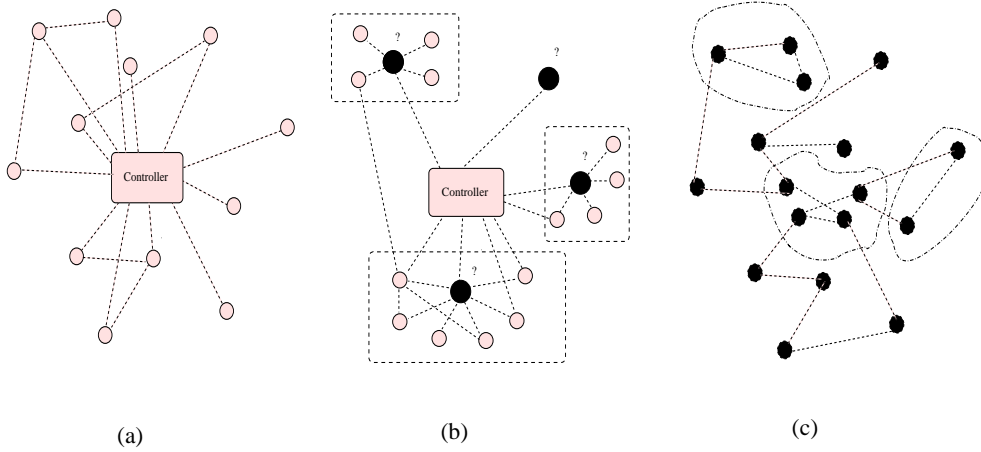


Figure 11: Three forms of management between a number of networked host nodes. The lines show a logical network of communication between the nodes. Dark nodes can determine their own behaviour, i.e. they have political freedom, while light nodes are controlled. The models display a progression from centralized control to complete de-centralization of policy (see refs. [40, 41]).

bottleneck, or a constriction of the workflow. The entire world cannot ring the central office in Ireland with their help-desk queries. It would cost too much, and it would place too much burden in one place.

Centralization might seem like a rational way to manage systems, but it results in *single points of failure*, i.e. places where the system is vulnerable to destruction by random or malicious failures.

The opposite of centralization is a completely peer to peer distribution of nodes, with communication channels in some meshed network with nearest neighbours. Each node can relay messages that need to traverse the network beyond its immediate horizons. This results in a cheap solution, but one in which each node has control over its own faculties. A different principal is needed to manage systems which collaborate voluntarily[38, 39].

There are six architectures between a star topology and peer to peer autonomy. These are described in refs [40, 41]. Three of these models are shown in fig. 11.

The logical and physical layers can both be described in these terms, with different consequences. Centralization has the effect of generating bottlenecks for both logical and physical layers of a system. Distribution is the only strategy for scaling, but its cost is in a lack of perceived order. Psychologically, many administrators feel uncomfortable with the idea of releasing the leash of control that one perceives in a centralized system. The actual uncertainty associated with any of the workflow models is, however, dependent on too many factors to be able to say that one method or the other is better.

14.1 Continuum approximation

To examine the scalability, we consider a simple rate approximation based on average flows. We assume a simple linear relationship between the probability of successful maintenance and the rates of communication with the policy- and enforcement-sources. This need not be an accurate description of reality in order to lead to the correct scaling laws[41]. Let us suppose that a change of configuration ΔQ is proportional to an average rate of information flow I , over a time Δt ; that is

$$\Delta Q = I \Delta t. \quad (45)$$

This equation says that I represents the time-averaged flow over the interval of time for which it acts. As we are interested in the limiting behaviour for long times, this is sufficient

for the job.

In the following we will use p to denote the average (over time, and over all nodes) probability that configuration management information flow (repair current) is not available to a node. This unavailability may come from either link or node unreliability. We can lump all the unreliability into the links (see above) and so write

$$p = (1 - \langle A_{ij} \rangle), \quad (46)$$

where $\langle A_{ij} \rangle$ denotes both time and node-pair average. Each node then can only receive repair current during the fraction $(1 - p)$ of the total elapsed time.

The policy maintenance can be transmitted and/or enforced at a maximum rate given by the channel capacity of the source. Thus, in case (a) the currents must simply obey Kirchoff's law:

$$I_{\text{controller}} = I_1 + I_2 + \dots I_N. \quad (47)$$

The controller current cannot exceed its maximum capacity, which we denote by C_S . We assume that the controller puts out a 'repair current' at its full capacity and that all nodes are average nodes. This gives that

$$I_{\text{repair}} = \frac{C_S}{N}. \quad (48)$$

The total current is limited only by the bottleneck of queued messages at the controller, thus the throughput per node is only $1/N$ of the total capacity.

As we move towards a peer model (c), the repair current becomes increasingly internalized in the nodes, and the need for concentrating the error currents from all the nodes at a single location is obviated. The models therefore tend towards a peer to peer architecture in which the number of nodes N becomes irrelevant and the error correction rate stabilizes to a constant value.

14.2 Reliability

A common definition of reliability is the *mean time before failure* (MTBF). This is the average amount of time we expect to elapse between serious failures of the system. Another way of expressing this is to use the *average uptime*, or the amount of time for which the system is responsive (waiting no more than a fixed length of time for a response).

$$\rho = \frac{\text{Mean uptime}}{\text{Total elapsed time}} \quad (49)$$

This is essentially a measure of the time-utilization of a device. Some like to define this in terms of the Mean Time Before Failure (MTBF) and the Mean Time To Repair (MTTR), i.e.

$$\rho = \frac{\text{MTBF}}{\text{MTBF} + \text{MTTR}}. \quad (50)$$

This is clearly a number between 0 and 1. Many network device vendors quote these values with the number of 9's it yields, e.g. 0.99999.

Ad hoc networks are networks whose adjacency matrices are subject to a strong, apparently random time variation. If we look at the average adjacency matrices, over time, then we can represent the probability of connectivity in the network as an adjacency matrix of probabilities. For our purposes, the important property of the ad-hoc net is the intermittency of the links, due to the components' mobility.

An ad hoc network is thus represented by a symmetric matrix of probabilities for adjacency. Thus the time average of the adjacency matrix (for, e.g., four components) may be written as

$$\langle A \rangle = \begin{pmatrix} 0 & p_{12} & p_{13} & p_{14} \\ p_{21} & 0 & p_{23} & p_{24} \\ p_{31} & p_{32} & 0 & p_{34} \\ p_{41} & p_{42} & p_{43} & 0 \end{pmatrix} \quad (51)$$

One can move the probabilities (uncertainties) for availability from the host vectors to the connectivity matrix and vice versa; for example:

$$\begin{pmatrix} p_1 \\ p_2 \\ p_3 \end{pmatrix}^T \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix} \begin{pmatrix} p_1 \\ p_2 \\ p_3 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}^T \begin{pmatrix} 0 & p_1 p_2 & p_1 p_3 \\ p_2 p_1 & 0 & p_2 p_3 \\ p_3 p_1 & p_3 p_2 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}. \quad (52)$$

This equivalence tells us that a given host in the network can always regard the failures in hosts as being failures within the communications channel, or network, between them (though not vice versa). Thus all collaborating networks of clients and servers (peers) are ad hoc, probabilistic networks.

14.3 Redundancy and Folk theorems

The only strategy certain against component failure in system is redundancy, or parallelism. Systems are a combination of serial dependencies and parallel flows. Several theorems exist showing the probabilities of failure in the presence of serialism and parallelism[1, 4]. The most important one, perhaps, can be stated without proof as follows:

LOW LEVEL COMPONENT REDUNDANCY IS NEVER A WORSE STRATEGY
FOR RELIABILITY THAN HIGH LEVEL SYSTEM REDUNDANCY.

In other words, a single computer with redundant components is almost always better, and never worse than several redundant computers.

This can be understood as follows. Consider the example, taken from the excellent ref. [4]. If one starts with a basic system, whose architecture is based on necessity, parallelism can be added to secure redundancy in effective and ineffective ways. The folk theorem tells us that low level redundancy is never worse than high level redundancy; this is seen intuitively in fig. 12.

14.4 Human error

How does one measure and quantify the likelihood of human error? One method is clearly to use the empirical approach of fault trees[5, 6]. Another interesting approach has been employed in ref. [42], using ideas from reliability theory applied to the task network[4].

In this work, the authors presume that human error is related to the complexity of the tasks they have to perform. Task complexity boils down to a number of things:

- Execution complexity – manual errors in implementation.
- Memory complexity – how many things does a person have to remember?
- Parallelism – how does time-sharing/multitasking affect a person?

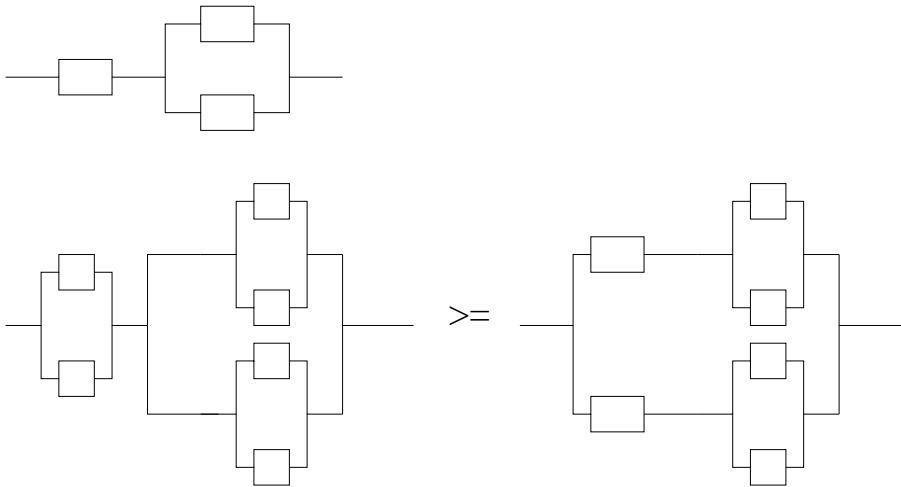


Figure 12: The upper architecture is parallelized in two ways below. In the left hand case redundancy is applied at the low level component layer. Disabling any single component results in the minimum disruption to the architecture. In the right hand solution, disabling the serial component from either the upper or lower arm results in the entire arm not working.

One can use estimations of complexity to argue that the probability of error is proportional to some function of these. The effect of errors in networks is a complicated issue around which a considerable body of theory exists[4, 1].

Task graphs can be viewed in two ways: in terms of open components that propagate input to output, or as larger closed systems of components with loops and cyclic connections. In the latter case, traditional network theory is less tractable. Newer methods such as network centrality can be useful tools for identifying vulnerable spots. using these methods on a task or dependency graph, one can find vulnerable spots and cut-sets quite easily.

15 Games - rational decision theory

The ability to model systems is an important step to understanding them, and therefore to being able to control them. Models that explain the causality of changing metrics of a system are called type I models[43]. Another kind of modelling that goes one step further is a type II model: this introduces a user-level, semantic interpretation of the system based on a set of values—i.e. it introduces a concept of policy, or user satisfaction. A type II model is about evaluating strategies for effectively achieving the goals of the system, i.e. planning and maintaining policy. Type II models consider the *utility* of decisions towards the goals of the system[44].

The theory of type II models is the theory of decision making: how to make rational decisions, i.e. how to justify conclusions by formalizing questions and answers. This is one of the ways in which we can optimize goals described by policy. Decisions which are motivated by goals are made *relative* to those goals so we must specify policy to eliminate arbitrariness in decision.

15.1 Classification of alternatives

Decision-making requires us to categorize or classify the available options. We can think of each category or option as a different symbol in an alphabet Σ , much as we did in section 9. Decisions can involve conflicting interests, between different individuals. How do these compete in with one another, in the eyes of a rational observer? i.e. What decision would a purely impartial observer make?

The simplest kinds of decisions are discrete yes/no Boolean decisions, which can be combined with operators AND/NOT/OR, as in section 11.4 (fig. 5). We know these from computer programming. They are a simplistic discrete view to decision representation, which is appropriate for a deterministic machine with a black and white view of the world.

Humans place different values on different kinds of issues, however. Continuous, non-deterministic decisions must be decided in terms of the value conferred to the decision maker, as measured in some form of currency. This is usually called a “payoff” or “utility”. Just as money, in markets, is only worth what people think it is worth – it has no absolute value – so the value of policy and decision is subjective and context dependent.

Placing a range of numerical values on decisions allows one to evaluate and describe when one situation is ‘better than’ or ‘worse than’ another. This is a simple matter in a utility viewpoint. To build up a similar structure in terms of logical assertions requires an extensive apparatus[45].

One can decide, for instance, that there is more than one “right” answer to a question. It is then necessary to make a value judgement about the ‘best’ of optimal answer. We can then weight the importance of the possibilities using the expected payoff a result yields back to us: with this approach, one can end up with a single optimal answer or a mixture of several answers that result in the biggest payoff or utility.

15.2 Mathematical Games

The theory of games, developed and generally attributed to von Neumann[44, 46, 47] is about competition between different goals in a common space. Opposing goals are thought of as different “players” in a game. which compete with one another to attack and defend one another’s positions. Two player games are generally sufficient for the kinds of models that can be dealt with in system administration. As with all models, there are discrete and continuous representations of game.

The discrete or extensive form of games involves making a decision tree of every possible action for every player in response to every possible move in a problem. The game tree is not unlike a fault or threat tree, described earlier (fig. 5). This level of detail is generally intractable from the viewpoint of modelling. As each decision confers some payoff to each player, ideally one would be able to add up the payoffs along each path, through the game, to find the best compromises at each branching point, given that both players are trying to do the same. The advantage of this approach is that one can retain the order of decisions. However, order does not always matter.

Such a two-player game is a tug-of-war between two players. Both players would like to win, and try their hardest, but they are prevented from having their own way by the other. The result is a compromise, or equilibrium.

The discrete modelling of the extensive form is difficult, so one generally approaches games from an averaged continuum approximation known as the strategic form of the game. This is also called the ‘normal’ form of the game. To reach this approximation, one lumps together actions that form single pathways into *classes* of play-behaviour for the players. It is somewhat analogous to high level management, rather than micro-management. The order of individual decisions is lost, as both players are now considered to act at the same time, in the manner of a duel.

Each player, in a strategic game, has a matrix Π of payoffs for playing a particular strategy, given the strategies played by the opponent (see fig. 13).

- Player 1 has alternative strategies: A, B, C .
- Player 2 has alternative strategies: α, β, γ .

If player 1 plays A and player 2 plays strategy α , then player 1 will receive payoff 1 and player 2 will receive payoff 3, and so on. Players can also use “mixed strategies” or linear

	A	B	C	
α	$\begin{array}{c} 1 \\ 3 \end{array}$	$\begin{array}{c} 2 \\ -2 \end{array}$		
β				
γ				

Figure 13: The twin payoff matrices for a strategic form game, written in a single table.

combinations of these basic possibilities:

$$S_1 = c_1 A + c_2 B + c_3 C + \dots \quad (53)$$

A particular admixture of strategies be viewed as being mixed during the course one game, or over several separate “plays” of the entire game.

An example game matrix $\Pi_{(1,2)}$ is shown below:

	Security hole	Bug in function	Missing function
Upgrade version now	(10,5)	(10,0)	(5,-5)
Test then upgrade	(5,5)	(3,9)	(0,8)
Keep parallel versions	(-10,5)	(-1,10)	(0,10)

The contents of the body of the matrix are the payoffs: these need to be chosen “rationally”. This choice or evaluation of utility is the crux of rational decision making. It is not the fact that one arranges them in a matrix, whether they are discrete or continuous values. The procedure for analyzing the payoff matrix could not be considered rational if the basis for the analysis were not itself rational: this evaluation of payoffs can often be subjective, so one has to be careful.

15.3 Making the optimal decision: solution methods

A solution of a game is a determination of optimal strategies for each player and a computation of the payoff-value received by each player when they behave rationally by trying to maximize their own payoff.

As both players are trying to maximize their payoff at the same time, the result will be a compromise. Saddle-points and other equilibria are of particular interest here.

Games are broadly classified into two kinds:

- If $\Pi_1 + \Pi_2 = 0$ a game is said to be a *zero sum game*, and the game is solved by von Neumann’s ‘minimax method’. This case is special because there is a ‘best answer’ when the game has a saddle-point, and otherwise there is a mixture or combination of answers that gives best result.
- If $\Pi_1 \neq -\Pi_2$ have to use more general equilibrium ideas – e.g. Nash equilibrium.

Zero-sum games are solved by virtue of the saddle-point, minimax theorem: take the payoff matrix for either one of the players (e.g. Π_1) and compare:

$$\begin{aligned} \max_1 \min_2 \Pi_1 &\equiv v_1 \\ \min_2 \max_1 \Pi_1 &\equiv v_2 \end{aligned} \quad (54)$$

If $v_1 = v_2$ then there is a pure strategy solution, i.e. a “best” answer. Else there is a mixture of strategies for both players that yields an equilibrium. For example, let

$$\Pi_1 = \begin{pmatrix} 1 & -3 & -2 \\ 2 & 5 & 4 \\ 2 & 3 & 2 \end{pmatrix} \quad (55)$$

Then

$$\max_{\uparrow} \min_{\leftrightarrow} \Pi_1 = \max_{\uparrow} \begin{pmatrix} -3 \\ 2 \\ 2 \end{pmatrix} = 2 \quad (56)$$

Subsequently,

$$\min_{\leftrightarrow} \max_{\uparrow} \Pi_1 = \min_{\leftrightarrow} (2, 5, 4) = 2 \quad (57)$$

Thus $v_1 = v_2 = 2$ and the relevant strategies are: $(\uparrow, \leftrightarrow) = (2, 1) \text{ or } (3, 1)$

Dominated strategies are a particularly easy way of reducing the number of rational options by elimination. This method works both for zero-sum and non-zero-sum games. Sometimes the payoff matrix can be simplified for both players if one of the players has a weak strategy, i.e. one that there is no reason to play because it performs worse than all others. e.g.

$$\begin{pmatrix} 50 & -10 & -20 \\ 20 & 40 & -44 \\ 0 & -10 & -40 \end{pmatrix} \rightarrow \begin{pmatrix} 50 & -10 \\ 20 & 40 \\ 0 & -10 \end{pmatrix} \quad (58)$$

The Nash equilibrium is the most popular solution concept for non-zero sum games. It is equivalent to minimax for zero-sum games. Computer software such as GAMBIT can be used to locate equilibria.

The Nash solution concept is based on the idea of bargaining – or economic agreement. If players can win by working together in the long-term, we call games cooperative, e.g. imagine two devices sharing wireless bandwidth either in a friendly way or competing aggressively to use the maximum share.

$$\Pi = \left(\begin{array}{c|cc} & \text{Compete} & \text{Share} \\ \hline \text{Compete} & ? & ? \\ \text{Share} & ? & ? \end{array} \right) \quad (59)$$

Does it pay them to share nicely or to compete aggressively? e.g. should you work with a competitor or compete and risk losing the fight? To get the whole picture, we need to model payoffs. This might be a complex mixture of policy-motivated expressions, joined into a common currency. This idea leads one to so-called *dilemma games*, of which the ‘prisoner’s dilemma’ is the most famous[48, 49, 50].

Consider a choice which involves making a promise to a competitor, such as delivering on a Service Level Agreement, or some other policy decision. The agreement has to be mutual, i.e. both players have something of value that the other wants, and thus they are in a bargaining position. Player 1 offers his promise to 2; player 2 offers her promise to 1. BGP peering relationships are of this type, for example.

Either party offering a promise can either ‘cooperate’ or ‘defect’ from the obligation implied by the agreement. What do we do if we have this? Let ‘C’ mean cooperate and ‘D’ mean “defect” from cooperation. The payoff matrix is given in fig. 14

- R is the reward for mutual cooperation.
- T is the value of a temptation to “defect”.

	C_1	D_1
C_2	(R,R)	(T,S)
D_2	(S,T)	(P,P)

Figure 14: The basic form of a dilemma game: a non-zero sum bargaining problem.

- S is the sucker's payoff, for getting exploited.
- P is the punishment for both losing the gamble of trust.

We assume $T > R > P > S$ and we see both players gamble on each others' kindness. The result is that:

- If both cooperate, both get maximum reward.
- If one cooperates and the other defects, the defector wins by exploiting the other.
- If both try to exploit each other, they both lose with a minimum payoff.

If we treat each game as a single interaction and play the game many times, so each player can only see *previous* rounds, then the strategy of 'tit for tat' is found to be most effective at stabilizing a players' returns. The dilemma scenario is used to argue for the significance of strategy in conflict/cooperation scenarios. There is a large literature on this topic.

The dilemma game is the basis for discussions of economics in the system. It is a representation of the maxim 'you scratch my back and I'll scratch yours', i.e. trade of services for mutual gain, such as peering in BGP. The game framework allows one to negotiate the equilibrium contract between the parties, motivated on rational grounds. This kind of analysis has been used in auto-discovery protocols, using auctions for electing network representatives amongst equal peers.

16 Conclusions

A mixed strategy of rationality and causality enables system administrators to formalize problems of planning and management, through a variety of methods. These methods make contact with diverse fields of research. Theory is an increasingly difficult pill to sell and swallow in this age of increasing mathematical illiteracy, and yet we need it more than ever. This lightning overview brings together only a few of the ideas required in network and system administration.

References

- [1] M. Burgess. *Analytical Network and System Administration — Managing Human-Computer Systems*. J. Wiley & Sons, Chichester, 2004.
- [2] J. Topping. *Errors of Observation and their Treatment*. Chapman and Hall, 1972.
- [3] R.H. Dieck. *Measurement and Uncertainty (Methods and Applications)*. Instrument, Systems and Automation Society, third edition edition, 2002.

- [4] A. Høyland and M. Rausand. *System Reliability Theory: Models and Statistical Methods*. J. Wiley & Sons, New York, 1994.
- [5] U.S. Nuclear Regulatory Commission NRC. *Fault Tree Handbook*. NUREG-0492, Springfield, 1981.
- [6] R. Apthorpe. A probabilistic approach to estimating computer system reliability. *Proceedings of the Fifteenth Systems Administration Conference (LISA XV) (USENIX Association: Berkeley, CA)*, page 31, 2001.
- [7] M. Burgess. Cfengine’s immunity model of evolving configuration management. *Science of Computer Programming*, 51:197, 2004.
- [8] M. Burgess. A site configuration engine. *Computing systems (MIT Press: Cambridge MA)*, 8:309, 1995.
- [9] S. Traugott and J. Huddleston. Bootstrapping an infrastructure. *Proceedings of the Twelfth Systems Administration Conference (LISA XII) (USENIX Association: Berkeley, CA)*, page 181, 1998.
- [10] M. Burgess. Computer immunology. *Proceedings of the Twelfth Systems Administration Conference (LISA XII) (USENIX Association: Berkeley, CA)*, page 283, 1998.
- [11] G.R. Grimmett and D.R. Stirzaker. *Probability and random processes (3rd edition)*. Oxford scientific publications, Oxford, 2001.
- [12] A. Couch and Y. Sun. On observed reproducibility in network configuration management. *Science of Computer Programming*, 53:215–253, 2004.
- [13] J. Strassner. *Police Based Management, Solutions for the Next Generation*. Morgan Kaufmann/Elsevier, 2004.
- [14] J. Strassner. A model driven architecture for telecommunications systems using denng. In *Proceedings of the 2nd International Conference on E-Business and Telecommunication Networks (ICETE)*, pages 118–128, 2004.
- [15] TM forum. <http://www.tmforum.org>, 2005.
- [16] C.J. Date. *Introduction to Database Systems (7th edition)*. Addison Wesley, Reading, MA, 1999.
- [17] R. David and H. Alla. Petri nets for modelling of dynamic systems — a survey. *Automatica*, 30:175–202, 1994.
- [18] J.F. Meyer, A. Movaghar, and W.H. Sanders. Stochastic activity networks: structure, behavior and application. *Proceedings of the International Conference on Timed Petri Nets*, page 106, 1985.
- [19] Tony Clark, Andy Evans, and Stuart Kent. A meta-model facility for a family of UML constraint languages. In Tony Clark and Jos Warmer, editors, *Object Modeling with the OCL: The Rationale behind the Object Constraint Language*, pages 4–20. Springer, 2002.
- [20] J. A. Sykes and P. Gupta. UML modeling support for early reuse decisions in component-based development. In Keng Siau and Terry Halpin, editors, *Unified Modeling Language: Systems Analysis, Design and Development Issues*, chapter 5, pages 75–88. Idea Publishing Group, 2001.
- [21] J. Bergstra, A. Ponse, and S.M. Smolka, editors. *The Handbook of Process Algebra*. Elsevier, 2001.

- [22] S.C. Graves, A.H.G. Rinnooy Kan, and P.H. Zipkin, editors. *Logistics of Production and Inventory*, volume 4 of *Handbooks in Operations Research and Management Science*. Elsevier, 1993.
- [23] M. Burgess, G. Canright, and K. Engø. A graph theoretical model of computer security: from file access to social engineering. *International Journal of Information Security*, 3:70–85, 2004.
- [24] M. E. J. Newman, S.H. Strogatz, and D.J. Watts. Random graphs with arbitrary degree distributions and their applications. *Physical Review E*, 64:026118, 2001.
- [25] H. Lewis and C. Papadimitriou. *Elements of the Theory of Computation, Second edition*. Prentice Hall, New York, 1997.
- [26] C.E. Shannon and W. Weaver. *The mathematical theory of communication*. University of Illinois Press, Urbana, 1949.
- [27] M. Burgess. System administration as communication over a noisy channel. *Proceedings of the 3rd international system administration and networking conference (SANE2002)*, page 36, 2002.
- [28] T.M. Cover and J.A. Thomas. *Elements of Information Theory*. (J.Wiley & Sons., New York), 1991.
- [29] W. Willinger and V. Paxson. Where mathematics meets the internet. *Notices of the Am. Math. Soc.*, 45(8):961, 1998.
- [30] V. Paxson and S. Floyd. Wide area traffic: the failure of poisson modelling. *IEEE/ACM Transactions on networking*, 3(3):226, 1995.
- [31] W. Willinger, V. Paxson, and M.S. Taqqu. Self-similarity and heavy tails: structural modelling of network traffic. in *A practical guide to heavy tails: statistical techniques and applications*, pages 27–53, 1996.
- [32] W.E. Leland, M. Taqqu, W. Willinger, and D. Wilson. On the self-similar nature of ethernet traffic. *IEEE/ACM Transactions on Networking*, pages 1–15, 1994.
- [33] J. Cao, W.S. Cleveland, D. Lin, and D.X. Sun. *Non-linear Estimation and Classification*, chapter Internet Traffic Tends Toward Poisson and Independent as the Load Increases. Springer, 2002.
- [34] K.I. Hopcroft, E. Jakeman, and J.O. Matthews. Discrete scale-free distributions and associated limit theorems. *Journal of Mathematical Physics*, A37:L635–L642, 2004.
- [35] R. Jain. *The art of computer systems performance analysis*. Wiley Interscience, New York, 1991.
- [36] R.B. Cooper. *Stochastic Models*, volume 2 of *Handbooks in Operations Research and Management Science*, chapter Queueing Theory. Elsevier, 1990.
- [37] J. Walrand. *Stochastic Models*, volume 2 of *Handbooks in Operations Research and Management Science*, chapter Queueing Networks. Elsevier, 1990.
- [38] M. Burgess and S. Fagernes. Pervasive computing management ii: Voluntary cooperation. *IEEE eTransactions on Network and Service Management*, page (submitted).
- [39] Mark Burgess. An approach to policy based on autonomy and voluntary cooperation. *Lecture Notes on Computer Science*, 3775:97–108, 2005.

- [40] M. Burgess and G. Canright. Scalability of peer configuration management in partially reliable and ad hoc networks. *Proceedings of the VIII IFIP/IEEE IM conference on network management*, page 293, 2003.
- [41] M. Burgess and G. Canright. Scaling behaviour of peer configuration in logically ad hoc networks. *IEEE eTransactions on Network and Service Management*, 1:1, 2004.
- [42] J. Hellerstein, K. Katircioglu, and M. Surendra. A framework for applying inventory control to capacity management for utility computing. In *Proceedings of the IXth IFIP/IEEE International Symposium on Integrated Network Management IM'2005*, pages 237–250. IEEE, 2005.
- [43] M. Burgess. On the theory of system administration. *Science of Computer Programming*, 49:1, 2003.
- [44] J.V. Neumann and O. Morgenstern. *Theory of games and economic behaviour*. Princeton University Press, Princeton, 1944.
- [45] B.F. Chellas. *Modal Logic: An Introduction*. Cambridge University Press, 1980.
- [46] M. Dresher. *The mathematics of games of strategy*. Dover, New York, 1961.
- [47] R.B. Myerson. *Game theory: Analysis of Conflict*. (Harvard University Press, Cambridge, MA), 1991.
- [48] A. Rapoport. *Two-Person Game Theory*. Dover, New York, 1966.
- [49] R. Axelrod. *The Complexity of Cooperation: Agent-based Models of Competition and Collaboration*. Princeton Studies in Complexity, Princeton, 1997.
- [50] R. Axelrod. *The Evolution of Co-operation*. Penguin Books, 1990 (1984).

System Administration and the Business Process

M. Burgess

November 20, 2005

1 Introduction

The integration of computer technology into a business process is an obvious goal for an organization or company that relies on information services. In this chapter we ask: is it possible to understand business modelling and analytical system administration in the same light? In other words, how do we bring a rational scientific method to bear on the problem of building a business around an Information Technology (IT) infrastructure, or building an IT infrastructure for an existing business?

To do this, we must view process of business modelling in the same light as the process of modelling the rest of the information system, and to relate these ideas to the practical issues that businesses have to contend with.

The human component of both businesses and human-computer systems requires us to consider both rational decision making (optimization based on objective criteria) and irrational choices (arbitrary choices made as policy for reasons that are not part of a scientific framework, e.g. ethics or image). For example:

- Standards of “best practice”. Why are they best?
- Modelling a business process as an information system.
- Logistics of production and inventory.
- Identification of relevant scales of operation.
- Capacity planning.
- Time management.
- Strategic thinking: activity patterns like type I models, and decision planning like type II models.
- Gauging risks and uncertainties.
- Minimizing risk occurrences.
- Finding policies as stable equilibria.

One strategy for integrating business thinking into network and system management is to view activities in terms of services, rather than as low-level protocol interactions. This has become a popular paradigm since the ITIL/BS15000[1] language of service provision and Service Level Agreements etc. See also the work in the Telemangement Forum with eTOM and DEN-ng[2, 3].

Although the individual details of procedures are important, there is a more overarching need to understand the logistics of combining such procedures in the environment of an economic motor. We wish to turn some of these loose ideas into more rigorous ideas so

that one can apply the techniques of scientific modelling and rational decision making to them.

Not all models that are related to aspects of a business are ‘business models’ in the fuller sense of the word. There are good reasons to separate off specialized issues and consider them separately. A business model claims to supply the *raison d’être* of a business, along with its capacity for long term profitability. Like a computer system, a business is a box surrounded by an environment with which it is interacting.

Modelling something as complex as a business in full is probably not a practical proposition, using today’s technology. Perhaps in the future one might model businesses as one today models the weather, but we are some way off having this technology today.

A key difference with computer management is that businesses tend to seek growth rather than stability.

We shall also look at inventory models, efficiency models and so on, which pertain to smaller and more specialized parts of the whole.

The plan for this article is:

- Modelling business as a human-computer system.
- Identifying the principle for optimization.
- Managing internals, such as inventory and configuration.
- Planning high volume services and controlling uncertainties.

2 Heuristic goals of business

Whether explicitly defined or not, every business has a *business model* that defines the services, costs and strategies that allows it to survive in a competitive environment.

Service delivery is a key part of a business model. IT services are increasingly a part of this, but the principles for IT service provision are similar to those for the provision of any service.

Goal 1 (Service design) *A business must define the scope and quality of its services, ensure that they are deliverable and manageable, at the right cost.*

It is normal to speak of businesses as being service-based, in this discussion. This does not mean that supply or manufacturing industries are excluded, only that one should imagine supply businesses as providing a service to their customers. Thus the service paradigm is a convenient umbrella for business concepts.

Goal 2 (Capacity Management) *What rates of service delivery are required and how can they be achieved with a planned infrastructure.*

- A Service Level Objective (SLO) is the level of ambition that a provider aspires to.
- A Service Level Agreement (SLA) is a formalized agreement, i.e. a contract, between a provider and a client.

Goal 3 (Business relationships) *What is the structure of the organization with respect to suppliers and clients? Outsourcing must also be taken into account.*

Goal 4 (Risk and Security Management) *What are the risks of the business failure, either as a result of IT or other disasters, and what is the cost of securing against them?*

Goal 5 (Logistics of production) *What physical and resource constraints are there to providing the service?*

Goal 6 (Arbitrary Business policy) *Decisions which cannot be made using a rational method of decision e.g. ethical or moral prerogatives.*

Quality control is a subject that applies identically to system administration and business administration. It requires a proper understanding of the operation of a system.

- A risk model for the system is required, so that one identifies the most likely threats to the success of the business or system (e.g. what happens if there is illness amongst key personnel, or a major power failure or natural disaster such as a flood?). Identifying these threats is important for the next point.
- Completeness of procedures: a response procedure should exist and be implementable for each event that can occur within the system. Responses to events should be mandatory and be executed within a minimum time-frame, determined by the scales identified above.
- Monitoring of performance: a non-intrusive method of monitoring performance should be available to all levels of the system or business, so that a natural feedback, with a minimum of overhead can be provided. System or Business Performance Indicators should address both internal procedures and external factors, which influence the markets for a business or the environment in which the activity of the system takes place.
- Securing security and efficiency in performance through redundancy, load balancing, resource management etc.

3 De facto standards for human procedure

Industry standards exist for setting basic quality assurances for IT business[1, 4]. The standards are often presented as existing in order to protect business relationships such as outsourcing, in which a company isolates and gives autonomy to a part of its operation. Standards cover issues such as:

- Management's role
- Documentation requirements
- Competence acquisition (training and hiring)
- Configuration management
- Location and scope
- Monitoring and review
- Service Level Agreements with clients
- Complaints procedures: Fault recovery, Incident management, Problem resolution
- Budgeting and inventory
- Business relationships: dependencies (outsourcing)
- Legal constraints

The IT Infrastructure Library (ITIL) was developed by the Central Computer and Telecommunications Agency of the British government and has been adopted by many companies. It attempts to provide a ‘cohesive set of best practices’ for businesses and organizations. A British Standard which summarizes the issues in ITIL is BS15000[1] (see the contribution in this volume by Scheffel.). The IT Service Management Forum (ITSMF) is now the custodian of the standard[5].

The enhanced Telecom Operations Map (eTOM) is claimed to be the most widely used and accepted standard for business process in the telecom industry. The eTOM is analogous to and overlaps with ITIL and describes business processes required by a service provider as well as how they interact. eTOM provides more of a top-down hierarchical view of an enterprise whose aim is to provide service to external customers. There is considerable overlap between eTOM and ITIL, as is testified by the eTOM/ITIL interpreters guide[6, 7], but the eTOM vision is motivated very much by the traditional telecommunications hierarchy. It attempts to describe an organization as a database.

Management frameworks have been discussed by several corporate research milieux[8, 9]. The authors emphasize the difference between IT Management (service level management) and IT Governance (strategic planning and integration of IT in the business process). Information models for business interactions have also been discussed. [10, 3]

4 Agreements, contracts and relationships

Contracts are a tool for both establishing goals and imposing penalties. They are a method of codifying voluntary cooperation between peers in a social network[11]. Service Level Agreements are examples of contracts for service provision between IT peers.

Determination and optimization these contracts mixed both human and technological issues[12, 13, 14]. In particular, the context sensitive nature of agreements, and the great variety of automated technologies that involve contract exchange motivates ontological methods of harmonization[15].

Modal (deontic) logic has been used as a framework for making language of contracts precise, in a legal context[16]. Often, one is more concerned with standardized service level agreements in which one quotes well-understood specifications for average response time, minimum service level etc.

In ref. [10] contracts are used to define the penalty costs for failing to live up to their contractual obligations. This is reminiscent of the economical model of promise breaking[17]. A continuum study of this kind of contractual penalty feedback between a pair of peers has been examined in ref. [18]. There the authors noted the potential instabilities in relationships between agents is they were allowed to impose punishments and penalties on one another.

A further example of policy motivated by contracts is promise theory theory[19, 20, 21]. Here one considers the stability and consistency of a series of agreements in a network, to see whether the sum of individual promises can lead to a consistent union of ideas.

5 System, Environment and Scales

In the previous chapter in this volume[22], systems are described in terms of their degrees of freedom and their constraints. One should look for the freedoms, constraints and natural scales of measurement that are involved in the operation of a system.

5.1 Freedoms and constraints

There is a plethora of ways of modelling freedoms and constraints for a business model. Needs analyses can tell us about the kinds of variables at our disposal. The activities of

customers and internal parts of the business are another way of identifying freedoms. Essentially the freedoms are the resources that are available to the enterprise, either directly or indirectly (e.g. by purchase or outsourcing).

Constraints on an enterprise include debts to be paid, sustainability or minimum profitability, physical and market limitations, equipment and processing etc.. “Use cases”, in the sense of software engineering, can represent small subsystems within the whole. A quantitative model must show how freedoms and constraints should be balanced.

Most managers would like to live in a world that was clockwork and deterministic, however this is far from the case. The identification of random processes is important because this signals the extent to which the larger environment impinges on business activities. Random processes are essentially processes which cannot be fully understood without a complete model of the ‘rest of the world’. Since such a model is impractical, one speaks of randomness, probabilities and expectations.

Event modelling is crucial in models and lead us to consider queueing systems, where service requests arrive as a stream of discrete ‘orders’ or units. Arrival and renewal processes are then needed to understand and optimize the strategies for dealing with demand[23, 24].

5.2 Scales

An identification of approximate numbers is the first step towards determining whether a business can sustain itself. The basic scales or numbers should be the average values, since most processes in a real world environment are stochastic in nature.

For example: what is the approximate demand for a service over a fiscal year? What is the rate of arrival of payments at different times of year? What are the estimated costs of running the business, e.g. the price per kilowatt hour of electricity, the rent of your data centre, wages for staff? What is the typical size and monetary value of a sale? How long does it take to complete and fulfill an order? How much profit can one make per order transaction? Given that short-term losses can be offset by long-term profits, over what time-scale could one afford to make a loss? Give that there might be inhomogeneous demand and supply, over what time scale are the prices of services and resources approximately constant?

Comparing time scales is particularly useful way of seeing whether one has adequate resources to carry out a task. An organization that cannot deliver on its goals will not survive long.

If one thinks in terms of queues and arrival processes, we can pose the questions about service processing in a suitably analytical way: what is the time scale of trends and fluctuations in the arrival of service requests[25]? How large are fluctuations about the mean values and what is the scatter and jitter in them?

What total number of transactions can one expect in an interval of time e.g. a year? (A common model of sparse events is the Poisson arrival model, but this is known to work only for very large numbers of sparse arrivals, within the measurement interval, in general.)

Finally, from queue theory one ought to be able to estimate, what is the average response time of a business to a customer, or a server to a client?

In these questions, we already see that the same issues are directly applicable in both businesses and computer systems. This should not be a surprise, since the operation of a business becomes essentially machine-like once it has been defined.

5.3 Risk modelling

Fault tree analysis[26, 27, 28, 29] is a way of modelling the deviations from goals. It is a kind of anomaly detection and associated response. Anomalies are categorized into a tree of likely occurrences, and the overall risk can be evaluated as a probability. After this, the result must be translated into loss of revenues etc, by embedding the probable failure in the

wider context of the income/cost model. What are the potential risks to loss of income or reputation? What is the contingency plan for recovery? The security management code of practice addresses this issue[4].

6 Business as a network service

The most important concept in a sustainable business process is that the value created by the process should be greater than or equal to the cost of operation.

6.1 A functional view of services

In a simplistic sense, a business model is something that explains the relationship between input and output. A business is a machine for generating wealth and produce (output), given a certain amount of investment and raw materials (input). The question is how it can be made self-sustaining (see fig. 1).

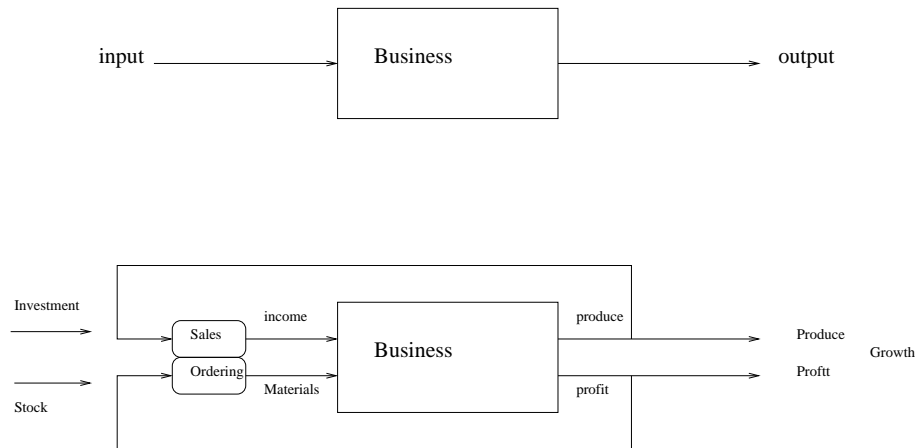


Figure 1: A functional viewpoint of a business as a self-sustaining process.

A self-sustaining process must have resources to begin with, and must produce more than it consumes. This is a peculiar idea, which seems to fly in the face of scientific reason and which readers may wish to ponder at length: how is it possible to sustain economic growth, and what is really being produced? Something from nothing? The answer, of course, comes down to understanding what value is, and why money is not a fixed scale of measurement, but rather a subjective convenience.

Let us set aside these issues and view the monetary value just as an arbitrary scale by which to express input and output. Then our simplest representation of a company is:

$$\text{output} = f(\text{input}) \quad (1)$$

which for the business translates in simplistic terms into:

$$\text{profit} = \text{BusinessValue}(\text{income}); \quad (2)$$

In terms of supply and demand, one inputs demand and outputs supply. This basic viewpoint of the business as a black box function with input and output is simplistic but essentially correct. There are various interpretations of this

Input	Output
income	payments
Materials	Production output

A more detailed modelling can be achieved by opening up and filling in the black-boxes, just as in software engineering.

$$(\text{profit}, \text{produce}) = \text{BusinessResult}(\text{income}, \text{materials}); \quad (3)$$

Then we take part of the profit to buy materials and sell the produce; thus functionally, we have a prototypical equation can be written in functional terms:

$$(\text{profit}, \text{produce}) \geq \text{BusinessResult}(\text{Income}(\text{produce}), \text{Materials}(\text{profit})); \quad (4)$$

We hope that this two dimensional mapping of produce and profit does not merely spiral to a fixed point of zero[30].

We do not normally control the input or output fully: it should be thought of as a random process[31]. Control factors such as advertising might influence expected demand at some predictable cost, but only within certain limits. Similarly production performance can be controlled by factors familiar in control theory[32], but only subject to basic limitations on certainty.

How shall we measure income? If the price of business services is constant then one could use either number of business transactions or the actual money paid into the business by external customers. But sometimes business give discounts for quantity etc, so there is a non-linear relationship between transactions and income.

Another issue is how should be coarse grain income? It has not traditionally been considered sensible to consider every individual micro-payment as a separate event. Rather one collects income over an interval such as a day, week or even month and considers the gross amount for that interval as the income. Similarly with payments, this coarse graining allows a certain smoothing of the dynamical processes involved, allowing temporary debts to accumulate and be covered within a planning horizon. We can simplify this by writing with fewer assumptions.

$$\text{BusinessResult}(\text{transactions}) \geq \text{Costs}(\text{transactions}) \quad (5)$$

By including costs in the function on the RHS we can also allow for the fact that costs might not be linear in respect of production. Transactions is now a more or less pure variable that measures the demand.

6.2 Deterministic and stochastic demand

Demand is a stochastic process — an arrival process. Indeed, this simple black-box model is a queue. We might imagine that the cost of the business process includes certain economies of scale and therefore falls off to some kind of asymptote with increasing demand. Similarly the result from a small number of sales might be small and rise to a maximum asymptote with increasing demand.

Using linear programming, one can see this on a simple planar diagram (see fig. 2). Any business enterprise needs to incorporate this basic framework. Timescales are important in this interpretation however (see fig. 3). It might be feasible to operate at a loss for short intervals of time, provided we make up for this loss later on.

The arrival of business opportunities is an essentially inhomogeneous random process which one should consider smoothing into trends to eliminate such fluctuations. By smoothing, or local averaging one can look at the trends and use these for seasonal planning[33].

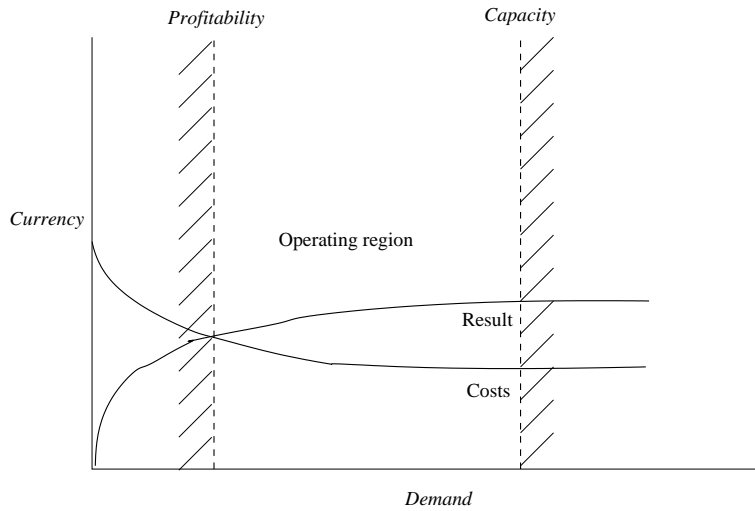


Figure 2: The operating region of a business lies in between profitability and capacity limits.

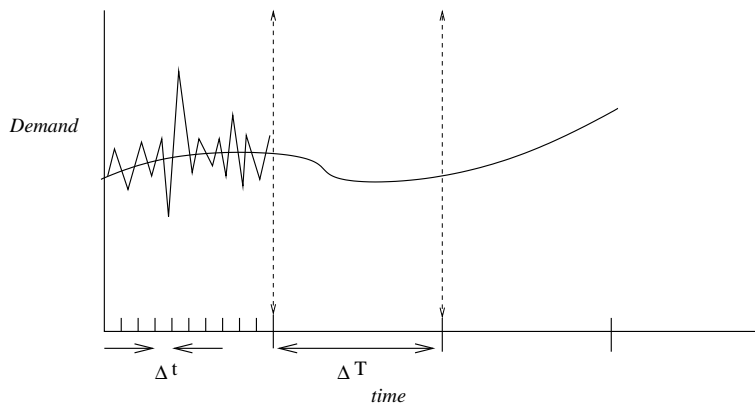


Figure 3: A time series view of the demand. The unpredictability of demand makes all its derived quantities random processes.

6.3 Diagrammatic modelling

The understanding of a system is greatly facilitated by diagrams and illustrations. Humans are particularly good at comprehending scenarios where several senses are involved. Visualization is one of the most important tools for comprehension.

Illustration generally begins by assembling the entities and relationships between them. There are many approaches to this, and one is unlikely to find a single diagram sufficient to describe what is happening in a business system. A more realistic expectation is to use a variety of diagrammatic techniques to model specific issues.

Two design strategies for building systems have emerged, and have developed with increasing refinements and compromises.

- *Top down*: The name ‘top down’ is motivated by the traditional way of drawing hierarchical structure, with high level (low detail) at the top, and increasing low-level detail at the bottom. A top down analysis is *goal driven*: one proceeds by describing the goals of the system, and by systematically breaking these up into components, by a process of *functional decomposition* or *normalization*.
- *Bottom up*: In a ‘bottom up’ design, one begins by building a ‘library’ of the com-

ponents that are probably required in order to build the system. This approach is thus *driven by specialization*. One then tries to assemble the components into larger structures, like building blocks, and fit them to the solution of the problem. This approach is useful for building a solution from ‘off the shelf’, existing tools.

The difference between these strategies is often one of pragmatism. A top down design is usually only possible if one is starting with a blank slate. It might not be possible to implement one’s wishes from a top-down viewpoint with an existing set of constraints and components.

A ‘bottom up’ design is a design based on existing constraints, namely the components or resources that are to hand. An advantage of building from the bottom up is that one solves each problem only once. In a top-down strategy one could conceivably encounter the same problem in different branches of the structure, and attempt to solve these instances independently. This could lead to inconsistent behaviour. The process of ‘normalization’[34] of a system is about eliminating such inconsistencies.

- (Computer system - ‘top down’) In the design of a new computer system, one examines the problem to be solved for its users (banking system, accounts), then one finds software packages which solve these problems, then one chooses a platform on which to run the software (Windows, Macintosh, Unix), and finally one buys the and deploys the hardware that will run those systems.
- (Enterprise - ‘top down’) In the organization of a maintenance crew, one looks at the problems which exist and breaks these down into a number of independent tasks (plumbing, electrical, ventilation). Each of these tasks is broken down into independent tasks (diagnosis, repair) and finally individuals are assigned to these tasks.
- (Computer system - ‘bottom up’) First one buys reliable hardware, often with operating system already installed, and installs it for all the users; then one looks for a software package that will run on that system and installs that. Finally, the users are taught to use the system.
- (Enterprise - ‘bottom up’) The enterprise bosses look at everyone they have working for them and catalogue their skills. These are the basic components of the organization. They are then grouped into teams which can cooperate to solve problems like diagnosis and repair. Finally these teams are assigned tasks from the list of plumbing, electrical work and ventilation.

In system administration, especially configuration management, these two strategies are both widely used in different ways. One may either implement primitive tools (bottom up) that are designed to automatically satisfy the constraints of a system, or one can use trial and error to find a top down approach that satisfies the constraints.

An entity-relation diagram, with associated data model, is a way of understanding information *types* and *relationships*, such as the approach used in the DEN-ng model[3]. However, such a diagram will not tell us anything about the efficiency of flows of work between the entities: for that, one needs something like a queueing network[24].

Transition diagrams provide a way of mapping out the activities in a system, in order to apply graphical methods of optimization[35].

This tells us how often the system spends its time the different states. By relating these states to the resources that are in use when the states are active, one can identify the most used resources during the interactive process. A centrality analysis of this graph can yield important information about resource allocation. For a deeper discussion of graph analyses, see the contribution by Canright and Engø-Monsen in this volume[36].

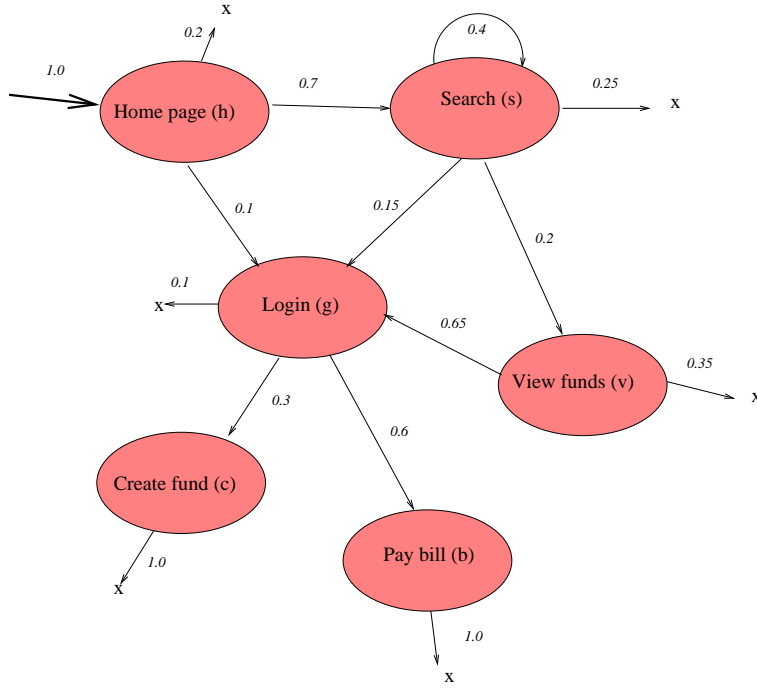


Figure 4: State transition diagram for a web based service. The transition probabilities weight the connections.

6.4 Continuum workflow models: scalability

Flow models take over where modelling of individual transactions becomes intractable. Continuous flows of information, goods or services are a convenient fiction for applying probabilistic methods. One replaces actual measurements with expectation values and events with probabilities.

Flow models use network or graph theory to analyse outcomes and consequences of flows. The six basic models of networking scalability are explained in [37]. See also the brief treatment in the foregoing chapter.

6.5 The hidden costs of ownership

A common way of expressing actual costs associated with components, in an enterprise, is the concept of *total cost of ownership*. Formally, this is the sum of all costs associated with the equipment, including the initial purchase price and licensing, over the lifetime of the equipment.

Let C be the number of customers, λ be the arrival rate of transactions, and P be the price of a transaction. Then we can say that, on average, the total income I , over a period is proportional to the product of these multiplied by the time interval Δt :

$$\text{Income } I \propto \langle C \rangle \langle \lambda \rangle \langle P \rangle \Delta t. \quad (6)$$

As usual, we use the notation $\langle X \rangle$ to mean the mean or expectation value of X . Thus business income from an online service will depend on the processing capacity[12]. Without proper modelling, it is easy to argue incorrectly that the correct strategy for equipment purchasing is to buy the fastest processor for the cheapest price. This is too simplistic, as we now show with an example.

Computer logic chips work by dumping large amounts of charge to ground through resistive materials, and because there is physical motion of mechanical components such as

disks and fans, almost all of the power consumption of a computer ends up as heat eventually. Power consumption of modern computers is a major expense. Moreover, the cost of cooling the equipment is also proportional to the total power output of the environment, and thus to the transaction rate of data processing.

Recently several large companies have boasted the use of off-the-shelf PC or Macintosh hardware for high performance computing servers. This sounds like an idealistic and attractive idea, but the economics of using cheap equipment in data centres are not as clear cut as one would believe. Measuring power consumption in a data-centre environment is a non-trivial task: power companies charge by delivered current, but that always over-estimates the actual power consumed, owing to *power factor* phase relationships in the alternating supply. Money can be saved even by investing in power factor correcting devices such as Uninterruptible Power Supplies, or more expensive hardware.

Inexperienced organizations think of the purchase cost rather than the cost of ownership, in production. Consider a simple example, based on numerical estimates from the Oslo region, here in Norway¹.

Let us suppose that we have a company considering populating a server farm with either cheap, off-the-shelf computer servers, or with high quality blade-racks.

Cost	Shelf Hardware	Quality Hardware
Purchase server	500 mu	1000 mu
Purchase switch	10000 mu	1000 mu
Power usage per server	0.1 kW	0.05 kW
Rent per 100 units per year	2400	600
Failures	15%	1%

One hundred servers, with associated switching equipment costs of the order of 60,000 mu for off-the shelf hardware and 101,000 mu for the quality hardware.

Electricity costs about 0.035 mu per kilowatt hour. Our cheap servers produce about 0.1 kilowatts of heat per unit, while the quality servers produce half this amount per unit. The cost of cooling is proportional to the heat produced. Thus it is proportional to the power consumption. Let us suppose then that cooling adds fifty percent to the total power consumption. Thus, in a data centre with a hundred servers, we are therefore paying electricity for a year, to the tune of

$$\begin{aligned} E_{\text{shelf}} &= 100 \times 0.035 \times 1.5 \times 0.1 \times (24 \times 365) = 4599 \text{ moneyunits} \\ E_{\text{quality}} &= 100 \times 0.035 \times 0.5 \times 0.05 \times (24 \times 365) = 2300 \text{ moneyunits.} \end{aligned} \quad (7)$$

Thus the 41,000 money units saved on purchase cost leads to 2300 money units per year of additional power costs. It would take 18 years to justify the additional cost of these servers, by this reckoning. However, one must also add to this the cost of storage, cooling, management and reliability (replacements and repairs).

Cheap off-the-shelf PCs are large and bulky and need storage and cabling, and additional switching equipment. In additional rented infrastructure, one can expect at least:

$$\begin{aligned} R_{\text{shelf}} &= 2400 \\ R_{\text{quality}} &= 600 \end{aligned} \quad (8)$$

Failure costs, equivalent to 15 of the cheap PCs per year occur, but only 1 of the more expensive machines fails, so

$$\begin{aligned} F_{\text{shelf}} &= 15 \times 500 = 7500 \\ F_{\text{quality}} &= 1 \times 1000 \end{aligned} \quad (9)$$

¹The costs are approximate and based on Norwegian kroner divided by a factor of ten for more ready comparison with dollars and Euros.

The total ‘saving’ by buying off-the-shelf hardware (Δ = quality minus shelf) is thus

$$\begin{aligned}\Delta S &= \Delta R + \Delta F + \Delta E \\ &= -1800 - 6500 - 2300 \\ &= -10600 \text{mu/year}\end{aligned}\tag{10}$$

In $41000/10600 = 3.9$ years, one has recovered the cost of the quality servers. This could be the actual lifetime of the cheap off-the-shelf PCs, thus, it is not clear that one actually saves any money by buying cheap commodity hardware.

We have not included the costs of administrating and lost revenue due to down-time etc. There are additional costs involved here which could tend to favour more expensive hardware. Soon, the rate of development in chip speeds etc will slow down and environmental levies will be placed on the disposal or recycling of computing equipment and this cost will also be paid by the consumer. Thus the present style of throwing away equipment after a few years will eventually have to change. Investment in quality could be the strategy of the future.

7 Demand and capacity: ‘provisioning’

The Americanization ‘provisioning’ has become common parlance for the activity of providing the necessary utilities to cope with demand, i.e. the provision of service capacity.

All we can say with certainty, about a system, is that the rate of production in the business cannot exceed some maximum capacity C . If a business is merely a fixed machine whose costs are fixed, then the capacity of the system can be fixed. The problem then is to maximize capacity to cope the maximum expected demand. (see fig 5) This is often done by arranging for a certain over-capacity. However, if costs are proportional to some monotonic function of capacity then one could envisage an enterprise saving money by adjusting their available capacity (re-deploying equipment, hiring and firing etc). See fig. 6.

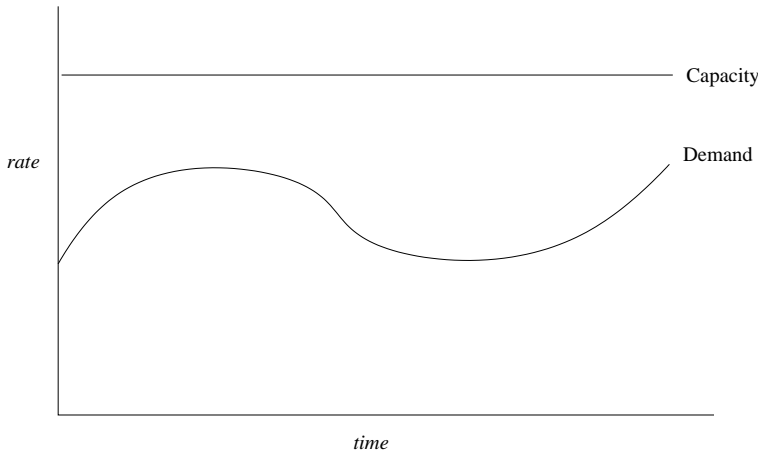


Figure 5: An over capacity—does this gap between capacity and demand cost anything for the company?.

8 Service provision models

Network services are the new paradigm for almost every new enterprise and service today. Even production industries can be regarded as services producing goods within a network

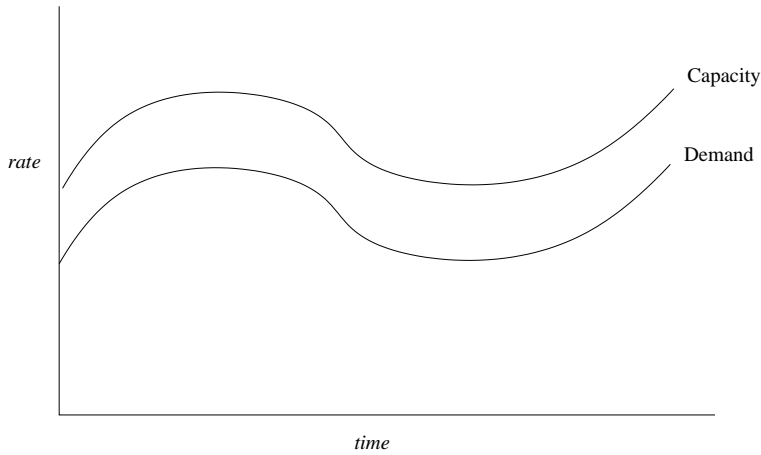


Figure 6: Capacity adjusted to follow demand—in principle more efficient.

of suppliers and consumers. The service model is simply a way of thinking. The advance which accompanied this point of view was the idea that delivery of a service is a contractual agreement between parties, and that customers are allowed to expect a certain Quality of Service (QoS).

Quality levels are negotiated using Service Level Agreements (SLA). In the literature, these concepts have multiplied into a plethora of Three Letter Abbreviations, including Service Level Objectives (SLO), Service Level Indicators (SLI) etc. A service level agreement is a contract, and thus one imagines that failure to live up to the expectations of the agreement will lead to reprisals, such as unpaid bills or lawsuits.

One is therefore interesting in dimensioning service capacity to be able to fulfill the contracts that have been signed. There are various approaches to this.

- Over-provisioning: by making service capacity much better than demand, one allows for any margins of error by always having extra capacity. This implies a certain level of waste.
- Optimization: calculating the optimal resource needs using a models and data measurements.
- Inventory management: use of inventory control methods to provide a more dynamic resource management based on periodic maintenance of stocks (this is somewhat analogous to periodic system maintenance regimes).

The service paradigm allows us to couple models to well-known analyses like queueing theory, which deals with everything from packet networks to vehicle traffic.

In terms of business, or enterprise organization we would like to have some way of calculating reasonable boundaries and levels for service, based on what we know of our capabilities. An interesting approach to this has been suggested by Sauvé et al. [12]. There one considers a Service Level Agreement to consist of two essential items per service type:

- A minimum service level (availability)
- An average response time $\langle t \rangle$

In addition two input values are required: a longest response before users ‘defect’, or give up on delivery and go elsewhere, and a probability distribution for the response time in relation to the threshold, so that the probability of being over threshold can be computed. Using this basic information, one can create a cost model for service provision, including

over-capacity. By minimizing a cost function, one can determine the optimum number of servers required to support a given Service Level Agreement, or the maximum values in an agreement that can currently be supported by current hardware.

8.1 Quality of Service

The concept of Quality of Service (QoS) has moved centre stage in network management as service providers seek to offer measurable guarantees to their customers in the form of Service Level Agreements (SLA). The Quality of Service builds on terms like QoD (Quality of Devices), QoE (Quality of Experience), QoB (Quality of Business), and any number of variations to discuss the issue of service provision. Each of these is trying to capture the essence of a usable measure, or ‘value’, that can be sold to customers. It has been suggested that service quality must be a function of ‘Quality of Devices’, since service is a function of devices:

$$\text{QoS} = f(\text{QoD}) \quad (11)$$

This makes clear sense: it follows from the laws of causality. The question remains, however, what kind of function should this be, and would a knowledge of the function help us to understand the limits of predictability of service levels?

Quality of Service advertises a qualitative characterization, where one would prefer a quantitative one. One must therefore pin down the meaning of the term.

The key measurable to be average service rate R , measured in *bits per second*. We interpret Quality of Service to mean the level of predictability in the *quantity of service*; i.e. we identify:

$$\text{Quantity of Service} \pm 1/\text{Quality of Service} \rightarrow R \pm \Delta R. \quad (12)$$

Thus, if quality of service is high, the uncertainty in service level is small and guarantees will be reliable; and if it is low, the uncertainty is high and guarantees are likely to be unreliable. This viewpoint ties Quality of Services to more traditional quantities, such as reliability and certainty.

Ref. [13] also attempts a causal approach. This paper captures the spirit of a causal approach by suggesting the link between service quality to its dependent variables, but it does not take the idea to its logical conclusion by actually defining the functional relationships or how they relate to the service quality. In ref. [38], a heuristic overview with similar ideas was presented, and in ref. [13] an acceptance of stochastic ideas was acknowledged in modelling using network simulations.

The method of Fault Tree Analysis[29, 26] can be used to construct an overview of the causal processes. The theory of errors and uncertainties[39, 40] may be used to estimate the probabilities.

Fault tree analysis can be used to gauge both the likelihood of error and the uncertainty in service provision (see fig. 7). Probabilities can be used to represent either the likelihood of a service failure (i.e. the likelihood that an SLA is not fulfilled), or the probability of deviation from expected (average) service levels. In the latter case, there this gives us a way of deriving the limits of certainty in service provision.

For instance, suppose we have – at the very lowest level of the system – that the data rate for the transmission medium is given by Shannon’s Gaussian noise formula:

$$C(B, S, N) = B \log_2 \left(1 + \frac{S}{N} \right), \quad (13)$$

where C is the channel capacity in bits per second, B is the bandwidth in Hertz, S is the signal power and N is the noise power. We now wish to discuss the effect of temperature

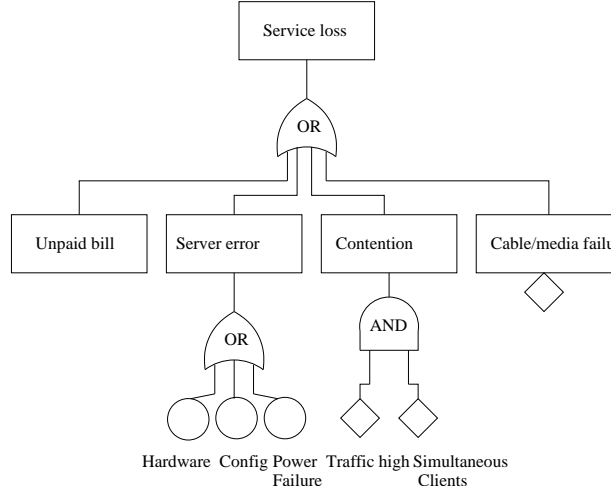


Figure 7: Fault trees can capture the essence of the hierarchical relationship between quality of components and final service. Here is a typical example of how fault trees are used to gauge reliability. Logical AND and OR gates reflect the interdependency of lower level components in a system.

on the capacity of a metallic conductor (copper wire), for which it is known that thermal noise has the form

$$N_{\text{metal}}(T) = N_0(T + 273), \quad (14)$$

where N_0 is a constant and T is the temperature in Celsius². The complete formula is given by straightforward substitution:

$$C(B, S, T) = B \log_2 \left(1 + \frac{S}{N_0(T + 273)} \right). \quad (15)$$

With the full expression capturing all of the dependencies in the dependency tree, through substitutions, the theory of errors and uncertainties can now be applied to the list of independent parameters.

8.2 Non-linearity: reactionary models of service dependency

Service provision is, in some contexts, thought of as the delivery of fixed-rate, policy-controlled transactions between willing parties. A contract of service, called a Service Level Agreement (SLA) documents the promise that the service provider makes to the client. The SLA is a form of policy for the server to uphold[41, 42]; what distinguishes it from other policy rules is that it is not deterministically enforceable, since the parties have no direct control over one another.

Any service provider's policy is subject to environmental uncertainties and, for that reason, the promise can never be more than an expectation or hopeful prediction of average service levels[43]. To verify that a promise has been kept, the client must monitor the service level. Service Level Agreements are often complex. They combine several measured values into composite metrics that are then compared to a template[13, 14, 44, 45, 46]. This has important implications for the stability of policy relationships.

²Note that the uncertainty due to thermal noise is not negligible in high capacity copper wires: it can represent tens of Megabytes that might be sold to a customer.

Feedback regulation of policy is an idea that has received more attention recently, due to interest in autonomous system operation; it has been discussed in a variety of circumstances[47, 48, 49, 50] for different service types ranging from web services to configuration management. Feedback regulation occurs regardless of whether it is carried out autonomously by machines or by humans, whenever there is a dynamical interplay between monitoring and SLA.

From the theory of dynamical systems, it is known that the combination of several dynamical variables using AND (multiplication) and OR (addition) in feedback loops can lead to ‘chaotic’ or non-linear behaviour. Such combinatoric ideas have been discussed to provide more expressive policies for service level agreements[13, 14, 44, 45, 46]; for example, in ref. [13], the authors discuss ‘aggregated Quality of Service parameters’ of the form:

$$S \text{ depends_on } (SS_1 \text{ AND } SS_2) \quad (16)$$

as well as more general rules like:

$$QoS_A(S) = f(QoS_B(SS_1) \text{ AND } QoS_B(SS_2)). \quad (17)$$

They are basing service quality promises on a function of measured values. However, there are problems with combining dynamical variables non-linearly on finite precision systems. It is known that such systems can spiral out of control, or simply ebb to zero for certain ranges of parameters (for a review, see [51]). It is therefore important to understand how such behaviour emerges in dynamically regulated policies and what can be done to secure stable operation.

In ref. [18], the authors attempt to answer the basic question: whether or not a reactive policy, based on mutual observation of several data points, can lead to stable behaviour or whether escalation of reprisals by the parties can spiral out of control. The studies made on continuum approximation models indicate that feedback regulation of SLA-like policies is a highly complex matter that must be parameterized with caution. Even simple non-linear policies have no stable fixed point that can be associated reliable service levels, other than the trivial case of no service. If two parties enter into a policy agreement of this type, they will either undergo a long fibrillation of changing behaviour, or they will spiral down to offering zero service levels. Thus the automation of policy should be wary of these issues.

9 Inventory models

Most traditional businesses involve the actual sale of goods which need to be stored in some kind of warehouse. The name *inventory* is used for stocks of revenue generating goods. For example, an Internet bookseller, Like Amazon, needs a warehouse to keep its stock, pending retail sale to customers. There is a cost involved in storing goods, both in terms of expected returns and in terms of warehouse rent. The longer inventory is sitting in a warehouse, the less profitable it can be.

However, even a computer centre needs certain supplies to function. Computers, monitors, mice, keyboards, CDROMs, hard-disks, network cables etc, are all examples of potential inventory that a system administrator should consider keeping as inventory.

There are three considered motives for storing inventory, rather than ordering on demand (or so-called Just In Time) models[52].

1. Transaction motive: by ordering large or predictable quantities of the merchandise, one can minimize transaction costs, or administrative overhead k .
2. Precautionary motive: Keeping inventory is a way of minimizing the risks of uncertainty. It provides a buffer against late deliveries or sudden demand.

3. Speculative motive: If the price of stock is expected to rise in the near future, pre-ordering and stockpiling can be a way to save money.

Modern computer systems do something similar in their ‘caches’, where the cost is not immediately monetary but a cost in time. Recently some authors have attempted to map inventory models onto service based computing models[9]. Although these issues are motivations for inventory models, the models themselves only account for the first of these reasons.

Inventory models are essentially primitive form of queueing theory, where the focus is slightly different. They were invented to deal with actual warehouses (see the reviews in volume of ref. [52] for lucid introductions) and the logistics of planning, but they can be applied to Internet commerce and also elementary transaction performance models for networking. The over-simplifications involved in the most basic models are usually tolerated precisely because of their simplicity.

9.1 Basic continuum model of averages

The basic model of inventory is known as the Economic Order Quantity (EOQ) model and assumes a completely deterministic process of predictable supply and demand. While this is clearly not true, by any stretch of the imagination, it is plausibly true over long times for average quantities. It can therefore be thought of as a ‘mean field approximation’.

We make the following assumptions:

1. The inventory of merchandise is all of a single type and stored at a single location.
2. We expect inventory to cycle continuously with a provision of service that never stops (a deterministic queue). So we ignore start up costs, and we are interested in the costs per inventory cycle-period, i.e. the time period T of the inventory cycle is a natural timescale for the model.
3. The model is deterministic, i.e. the rate of consumption (or the *expected demand*, in a sales interpretation) for merchandise is known with certainty, and is λ inventory units per unit time. It is like a leaking bucket.
4. One cycle of the model is one single (average) order of (average) size q , taking (average) time t .
5. Shortages of inventory do not occur, nor does over-production, i.e. demand is always matched to supply, on average.
6. Delivery time is instantaneous, i.e. immeasurably short compared to an average timescale T for ordering.
7. We do not take into account the provision for discounts for old stock, i.e. merchandise is as valuable today as it was when it entered the inventory.

As always, the assumptions underpinning a model are the key to knowing how and when we should take its results seriously.

Let us assume that each order of goods is made with average quantity size q , which is clearly constant over the time interval for which the average is defined. The cost of each order, of size q , has a fixed average transaction price k (money units) for and an overhead that comes from the cost rate of *holding* the merchandise in the warehouse h (money units per unit time) over time.

Finally, let the actual inventory, as a function of time be written $I(t)$ (see fig. 8). From our constraining assumptions, the inventory function can only have the simple form:

$$I(t) = q - \lambda t, \quad (0 \leq t \leq T). \quad (18)$$

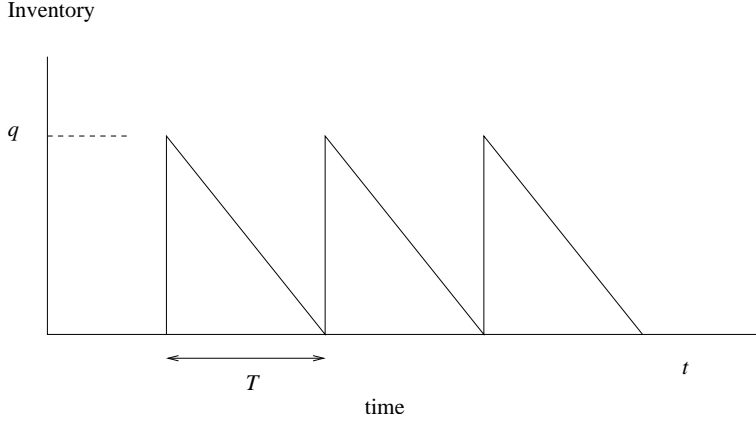


Figure 8: The basic inventory model with continuous cyclic operation shows how the stock is produced ‘suddenly’ and used at a constant rate λ until it is exhausted.

The cost per unit time of keeping merchandise is

$$C(T) = \frac{1}{T} \left(k + h \int_0^T I(t) dt \right) \quad (19)$$

and it measured in dimensions of money units per period. From fig. 8, we note that $T = q/\lambda$, just from the geometry of the triangles. This integral can be calculated trivially, or one can use the half-base-times-height rule for the area of a triangle:

$$\begin{aligned} \int_0^T I(t) dt &= (qT - \frac{1}{2}\lambda T^2)/T \\ &= \frac{1}{2}q^2\lambda \\ &= \frac{1}{2}qT. \end{aligned} \quad (20)$$

Thus, we have a cost per cycle/order function that may be expressed either as a function of q or of T :

$$C(T) = (k/T + \frac{1}{2}h\lambda T) \quad (21)$$

$$C(q) = (k\lambda/q + \frac{1}{2}hq). \quad (22)$$

In either variable, we see that the cost per order is a non-linear function that is the superposition of a hyperbola and a straight line (see fig. 9).

The cost function has a minimum, which may be found by differentiation by q or T , at the values q^* and T^* respectively:

$$q^* = \lambda T^* = \sqrt{2k\lambda/h}. \quad (23)$$

at which point

$$C(q^*) = C(T^*) = \sqrt{2k\lambda h}. \quad (24)$$

Thus, if we have optimized the average performance of this systems, the average cost per order grows with fixed order price with the holding cost.

The result displays a symmetry between order cost and holding price and demand:

$$\langle k, \lambda, h \rangle \quad (25)$$

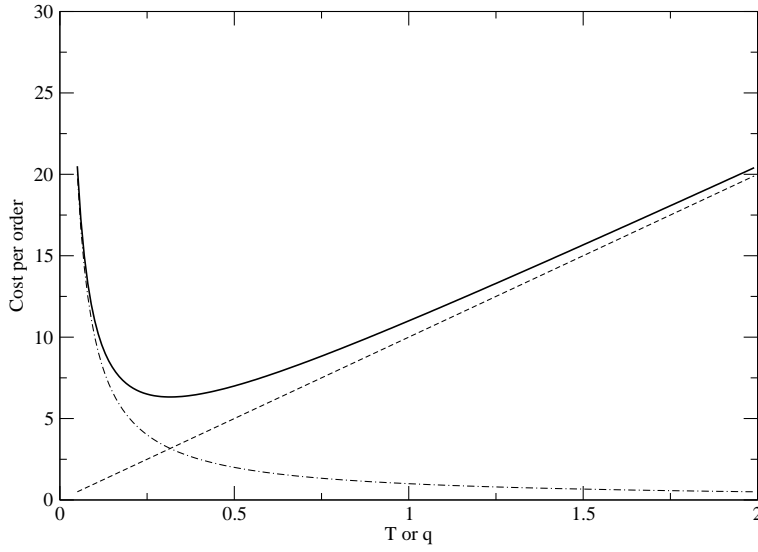


Figure 9: The solid line cost per order (or per cycle) function, is the superposition of a linear growth and hyperbolic cost per unit time. In either parameterization, has the form of a lifted hyperbola. There is a clear minimum at an abscissa point T^* or q^* , which represents the minimum cost value.

How do we explain this symmetry causally, and what does mean?

We must first explain the interpretation of the variables, and understand their forms.

- T^* is the cheapest time interval for customers to place their orders. If we make T less than this, eqn. (19) tells us that the transaction price will dominate the cost, i.e. the customers will pay the basic processing fees too often. If we make the time larger than this, the holding time dominates the cost, as the storage cost of the inventory increases and is passed on to the customer.
- q^* is the cheapest order size. It is related to the time by the constant demand rate, so they are not independent variables. If q exceeds the optimum, i.e. customers greedily order too much at a time then the order period increases, and vice versa.

The optimum cost increases as either the unit cost, storage cost or demands increase. This is because the larger costs force the consumer to slow down the rate ordering (increase T^*).

The steady-state, average assumptions of the model allow us to eliminate time as a variable. At optimal operation.

How do we know that we can tune the system to perform optimally? On average we can imagine that it is reasonable to compensate for non-optimal during a period by suitable monitoring.

Suppose we do not manage to compensate, a deviation is characterized by the beautiful dimensionless relation:

$$\frac{C(q)}{C(q^*)} = \frac{1}{2} \left(\frac{q}{q^*} + \frac{q^*}{q} \right). \quad (26)$$

Suppose we achieve $q = \frac{1}{2}q^*$, then this ratio is 1.25. In other words, missing the target by a hundred percent of the achieved goal leads to only a twenty-five percent increase in costs. This is quite insensitive.

9.2 Applying the model

The inventory model is not a complete business model, in the sense of dealing with income and expenditure. It is a sub-model that deals with optimizing the costs of holding stocks and supplies. The burden of applying these formulae, as always, is in finding reliable and realistic data for the inputs.

The optimal rate of ordering is not about how much the inventory costs to purchase. It is about the administration costs. This, if P is the purchase price per unit for inventory, the total cost of the order is simply qP , but the cost of ordering and keeping the inventory $C(q)$ is an overhead which is to be minimized.

- k is the administrative cost of placing an order, including the actual time taken converted using the man-hour rate.
- h is the cost of keeping inventory which includes storage, insurance, any interest paid over the period on money borrowed etc.

1. Suppose that the yearly consumption of PCs is about 20 per year. An order of PCs, which includes decision-making, discussion, and processing accounts for 5 man-hours of time at 50 money units per hour, hence $k = 250$ money units.

The cost of storage for these PCs, including cleaning and insurance per unit, per year is $h = 100$ money units per unit per year.

The optimal order size is now,

$$q^* = \sqrt{\frac{2 \times 250 \times 20}{100}} = 10 \text{ units} \quad (27)$$

and

$$T^* = \frac{1}{2} \text{ years.} \quad (28)$$

Thus, two purchases of ten units, over a year is the optimal solution.

2. For a second example, consider the consumption of disk space on a mail server. As users accumulate E-mail, or as an ISP accumulates customers with fixed quotas. (In fact dealing with fixed quotas is much easier to analyze because we can more easily estimate the consumption, without having to take into account how much users tidy up old mail etc).

We shall take disks to be the unit of inventory. Suppose the expected number of new customers (consumers) per year is 100 per year. If we assume that a typical hard-disk can accommodate 100 users, then this amounts to a usage of $\lambda = 1$ disk unit per year.

The processing cost for an order of disks is a half man-hour at 50 money units per hour, i.e. $k = 25$.

The holding cost for disk space (assuming that they are installed on arrival) includes power consumption, replacement disks on error etc. Suppose this amounts to a small amount of power consumption, plus a ten percent chance of having to buy a new disk at 500 money units. Let's take $h = 55$ money units per disk per year as the holding price. In fact, if we allow for the rapidly increasing capacity of disks, we might make h larger to account for the loss of disk space from ordering too early.

The optimal order size is now:

$$q^* = \sqrt{\frac{2 \times 25 \times 1}{55}} = 0.95 \text{ units} \quad (29)$$

This tells us that there is no benefit to mass ordering disks before they are needed. It is okay to order them one at a time, since they are relatively expensive to own, and relatively cheap to order.

9.3 Extended model

In reality, the production of merchandise takes a certain amount of time. to produce or assemble the inventory. We can make a trivial modification of the model to include a finite and linear production schedule, with the simplifying assumption that the rate of production ψ is always great than the rate of consumption λ .

We shall once again assume an order size of q and a cycle time of T . By analysing the

Inventory

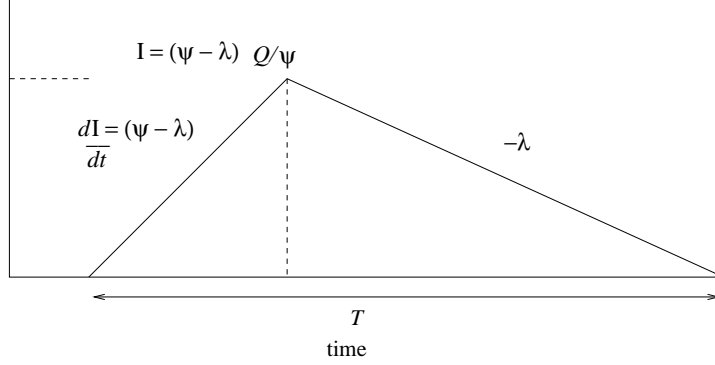


Figure 10: The extended inventory model with continuous cyclic operation shows how the stock is produced at a finite rate ψ and used at a constant rate λ until it is exhausted.

triangular period in fig. 10, we can use the triangle formula to find the area, as before in the integral of eqn (19). The vertical height of the triangle (representing the peak amount of inventory in store, given the stock is being used as it is being produced) is

$$I_{\max} = (\psi - \lambda)q/\psi. \quad (30)$$

The length of the base is still T . Moreover, by examining the triangle, we may confirm that $T = q/\lambda$ still holds.

The cost per order function is thus now:

$$\begin{aligned} C(q) &= (k + \frac{1}{2}h(\psi - \lambda)Tq/\psi)/T \\ &= \frac{k\lambda}{q} + \frac{1}{2}h(\psi - \lambda)\frac{q}{\psi} \end{aligned} \quad (31)$$

Minimizing this cost function, by differentiation gives an optimal order size:

$$q^* = \sqrt{\frac{2k\lambda\psi}{h(\psi - \lambda)}}, \quad (32)$$

and we recall that $\psi > \lambda$. As $\psi \rightarrow \infty$, this model become the basic model.

9.4 Effect of uncertainty in the rates

Even in an averaged regime, one must expect a certain amount of uncertainty in the rates of production and consumption. Suppose we assume that ψ and λ are independent variables subject to small, symmetrically distributed uncertainties:

$$\begin{aligned} \lambda &\rightarrow \lambda \pm \Delta\lambda \\ \psi &\rightarrow \psi \pm \Delta\psi \end{aligned} \quad (33)$$

Then, we may calculate the compound uncertainty in q^* by the Pythagorean combination of the first order Taylor expansions, in the usual way.

$$\begin{aligned}\Delta q^* &= \sqrt{\left(\frac{\partial q^*}{\partial \lambda}\right)^2 (\Delta \lambda)^2 + \left(\frac{\partial q^*}{\partial \psi}\right)^2 (\Delta \psi)^2} \\ &= \frac{1}{2} \frac{q^*}{|\psi - \lambda|} \sqrt{\left(\frac{\Delta \lambda}{\lambda}\right)^2 \psi^2 + \left(\frac{\Delta \psi}{\psi}\right)^2 \lambda^2}\end{aligned}\quad (34)$$

There is a similarity between the idea of inventory control and the idea of system maintenance. The maintenance theorem of ref. [43] says that a system is maintainable if

$$\frac{1}{\langle q \rangle} \frac{d\langle q \rangle}{dt} \ll \frac{1}{T} \quad (35)$$

where $\langle q \rangle$ is the average level of inventory.

However, we note that the inventory models are steady state models, so the maintenance theorem is satisfied automatically, since $\langle q \rangle$ is constant, by assumption. If, however, we allow for stochastic uncertainties in the model, then eqn. (35) becomes essentially

$$\frac{1}{q} \frac{\Delta q}{T} \ll \frac{1}{T} \quad (36)$$

which translates into $\delta \lambda / \lambda \ll 1$ and $\delta \psi / \psi \ll 1$.

10 What is the cost of poor system administration?

There are clearly many areas in which a system administrator's decisions affect the potential for a business to operate. In purchasing decisions and operating costs, there are large amounts of money to be saved. In a globalized environment, in which one has data centres with load balancing, companies could even consider re-routing requests to locations where load is low or power is cheap. Resource utilization management is an important part of cost management.

A competent engineer is one of the most important accessories to any machine or system. It is usually the member of personnel who is most forgotten until needed. Delays in an enterprise's operations could mean lost revenue, lost business or even lost reputation (an ability to attract new income). The repercussions can be serious.

In the present world of virtual issues like CPU speeds and numbers of megabytes, one sometimes forgets the importance of physical issues. What is the effect of heat? How should one avoid earth loops in a server room? Don't hang the leaky air-conditioner over your expensive mainframe, and so on.

10.1 Importance of empiricism

In the second world war, pilots were said to fly by the seat of their pants, meaning that they were intimately connected to their machinery—that they had learned the bumps and foibles of the machine. An understanding of the empirical nature of a system is just as important today, even if there are no physical vibrations.

How does a system respond under load? How much power does it use? How does the weather affect the performance of computing equipment, or the staff? While these might seem like frivolous questions, they have a very serious side.

In a data centre, the questions above become design issues. A system might thrash and slow down under heavy load. A peak of power consumption might cause fuses to blow or cooling equipment to overload. The weather can affect the air in the centre: if the humidity

is too high, condensation can form on cool surfaces; if it is too low (very cold weather), static electricity can cause errors and even power spikes that can damage equipment.

Perhaps we might think that running a server room a few degrees hotter can save some money in cooling power? Or that extra humidity can increase the specific heat capacity of the air and increase the efficiency of the heat transfer? These things can only be understood through rational inquiry and measurement. For example, the specific heat capacity of dry air is 1005 Joules per kilogramme per degree Kelvin, and the heat capacity of air with one hundred percent humidity is 1030 J/kg/K. The difference is only one percent, which is probably far less than normal power fluctuations, and is therefore irrelevant. On the other hand, humidity should be kept between forty to fifty percent to balance the risks of condensation and static electricity.

The scientific method: rational inquiry is a powerful medicine against such problems.

10.2 Return on investment and total cost of ownership

Return on investment is defined as the income per fiscal year, divided by the cost of the investment. This is often proportional to the throughput of the enterprise. Ineffective use of resources (poor scheduling or load balancing) can limit this. Poor training of staff and inadequate management can also affect it. Automation of basic tasks can increase the up-time of the system and avoid security problems.

10.3 Risk minimization

Understanding the risks of an organization and being able to plan for them is only possible with analytical tools. The identification of points of failure in the system network, the analysis of dependencies and probability that service targets will be met: these are examples of issues that are impossible to plan for without a mixture of empiricism and analysis.

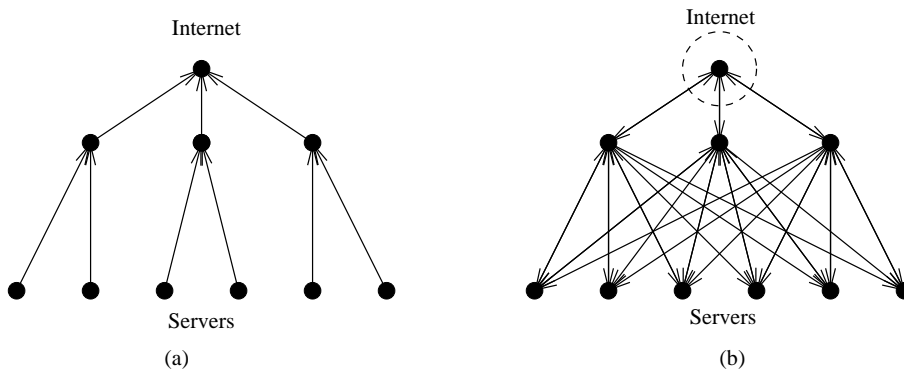


Figure 11: Load balancing without points of failure is a tricky matter if one is looking for redundancy. A load balancer is a tree – which is a structure with many intrinsic points of failure and bottlenecks. How can these be completely eliminated to make a high availability data centre?

Risk can be used as a means of optimization. For example, in ref. [53], the authors use a minimum risk principle to find the best time to make a backup copy of data in a busy data centre. Many administrators will do this when the system is most quiescent; however, an analysis of risk shows that that is not the way to minimize risk of loss due to random error. Rather, the answer is about halfway between peak activity and minimum activity (after peak time), since more changes that can be lost occur during peak times (see fig. 12). Whether an optimal solution is desirable is a matter for policy.

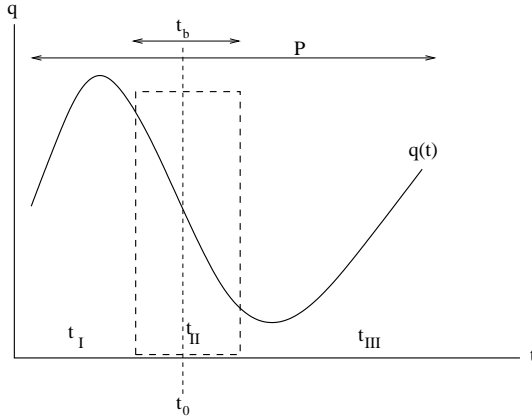


Figure 12: The ‘windshield wiper model’ for disk backup scans across files in a time t_b about t_0 . The wavy line shows the average rate of point change arrivals during a 24 hour period. There are three regions of currently unknown sizes: region I before backup, region II during backup and region III after backup. The file change process $q(t)$ brings q arrivals at time t and provides a probabilistic weight to the expected risk: if more files are arriving, the risk of loss will be higher.

10.4 Incident response and fire-fighting

Incident response is like an inventory model or queueing process. Incidents or events form an arrival, point process that accumulate into a queue or ‘inventory’ of tasks to be cleared. By analogy with the language used in configuration management policy, a business ‘incident’ could be defined as a deviation from policy, or it could be a complaint or a failure of some part of the business to deliver on a promise. We can model this kind of process with various levels of sophistication.

The illustration in fig. 8 can also be interpreted, somewhat simplistically, as a deterministic expectation of maintenance costs. At the leading edge of the saw-tooth one can imagine a quota of jobs to be complete arises (suddenly). Gradually, these jobs are completed and the inventory is cleared. After a periodic interval, one reschedules the next batch.

Although simplistic, this could be a realistic approach to time management—a way of clearing processes from a help-desk message queue, or some other E-mail queue. One simply chooses $\lambda = 1/T$ to fix the frequency of turn-around. There is an initial logistical cost of scheduling time to answer job messages (e.g. learning and research time to study the problem), and there is a cost associated with having them unanswered, or un-repaired (broken promises, maintenance contract penalties).

The optimum time T^* tells us how to prioritize this process; the optimal inventory order tells us how many jobs q^* to schedule at a time. This is related to the queue length in queueing models.

Another example of this kind of scheduling is periodically scheduled maintenance tasks, like garbage collection and configuration maintenance. where the scheduling order cost is the amount of resources needed to locate and fix potential problems. The holding cost is harder to estimate, but involves lost productivity or due to faults.

Incident management in relation to ITIL and business objectives is advocated in ref.[8]. The authors consider a policy-based prioritization, based on the penalty costs of failing to meet obligations in Service Level Agreements, and using time-discounting as an importance ranking, based on the urgency of the incident. It is a form of anomaly detection. These authors take essentially the analogous view to anomaly definition as is put forward for system administration in refs. [43, 54], namely that policy is an expression of the limits of a probability distribution. Prioritization can be defined essentially by the number of standard deviations off ‘normal’

Policy involved defining a series of mappings from low level events into a policy model. Thus policy involves an associative map, based on a notion of average correctness and a classification of incident ‘distance from normal’.

11 Conclusions

How does system administration integrate into the business process? It is centre stage. The system administrator plays the role of a service provider and of a maintenance agent. In a world of increasing reliance on computing systems, the engineers who design, tune and maintain the system are the keys to its success.

References

- [1] British Standards Institute. *BS15000 IT Service Management*, 2002.
- [2] Telemanagement Forum.
- [3] J Strassner. *Police Based Management, Solutions for the Next Generation*. Morgan Kaufmann/Elsevier, 2004.
- [4] British Standard/International Standard Organization. *BS/ISO17799 Information technology – Code of practice for information security managementt*, 2000.
- [5] Information Technology Service Management Forum. <http://www.itsmf.com>.
- [6] Telemanagement Forum. An interpreter’s guide for etom and itil practitioners, 2004.
- [7] J. Huang. [http://www.bptrends.com/publicationfiles/01-05 etom and itil - huang.pdf](http://www.bptrends.com/publicationfiles/01-05%20etom%20and%20itil%20-%20huang.pdf), 2005.
- [8] C. Bartolini and M. Sallé. Business driven prioritization of service incidents. (unpublished), 2005.
- [9] J. Hellerstein, K. Katircioglu, and M. Surendra. A framework for applying inventory control to capacity management for utility computing. In *Proceedings of the IXth IFIP/IEEE International Symposium on Integrated Network Management IM’2005*, pages 237–250. IEEE, 2005.
- [10] M. Sallé and C. Bartolini. Management by contact. In *Proceedings of 8th Network Operations and Management Symposium NOMS 2004*. IEEE/IFIP, 2004.
- [11] R. Axelrod. *The Evolution of Co-operation*. Penguin Books, 1990 (1984).
- [12] J. Sauvé et al. Sla design from a business perspective. In *IFIP/IEEE 16th international workshop on distributed systems operations and management (DSOM)*, in LNCS 3775.
- [13] G.B. Rodosek. Quality aspects in it service management. *IFIP/IEEE 13th International Workshop on Distributed Systems: Operations and Management (DSOM 2002)*, page 82, 2002.
- [14] D. Daly, G. Kar, and W. Sanders. Modeling of service-level agreements for composed services. *IFIP/IEEE 13th International Workshop on Distributed Systems: Operations and Management (DSOM 2002)*, page 4, 2002.
- [15] D. Trastour, C. Bartolini, and C. Priest. Semantic web support for the business-to-business e-commerce lifecycle, 2002.

- [16] A. Daskalopulu and M. Sergot. The representation of legal contracts. *AI & Society*, 11:6–17, 1997.
- [17] J.D. Carrillo and M. Dewatripont. Promises, promises. Technical Report 17278200000000058, UCLA Department of Economics, Levines’s Bibliography.
- [18] K. Begnum, M. Burgess, T.M. Jonassen, and S. Fagernes. Summary of the stability of service level agreements. In *Proceedings of International Policy Workshop 2005*.
- [19] Mark Burgess. An approach to policy based on autonomy and voluntary cooperation. *Lecture Notes on Computer Science*, 3775:97–108, 2005.
- [20] M. Burgess and S. Fagernes. Pervasive computing management i: A model of network policy with local autonomy. *IEEE eTransactions on Network and Service Management*, page (submitted).
- [21] M. Burgess and S. Fagernes. Pervasive computing management ii: Voluntary cooperation. *IEEE eTransactions on Network and Service Management*, page (submitted).
- [22] M. Burgess. *Handbook of Network and System Administration*, chapter System Administration and the Scientific Method. Elsevier, 2007.
- [23] R.B. Cooper. *Stochastic Models*, volume 2 of *Handbooks in Operations Research and Management Science*, chapter Queueing Theory. Elsevier, 1990.
- [24] J. Walrand. *Stochastic Models*, volume 2 of *Handbooks in Operations Research and Management Science*, chapter Queueing Networks. Elsevier, 1990.
- [25] J.P. Bouchard and M. Potters. *The Theory of Financial Risks*. Cambridge University Press, Cambridge, 2000.
- [26] U.S. Nuclear Regulatory Commission NRC. *Fault Tree Handbook*. NUREG-0492, Springfield, 1981.
- [27] A. Høyland and M. Rausand. *System Reliability Theory: Models and Statistical Methods*. J. Wiley & Sons, New York, 1994.
- [28] M. Steinder and A. Sethi. A survey of fault localization techniques in computer networks. *Science of Computer Programming*, 53:165, 2003.
- [29] R. Apthorpe. A probabilistic approach to estimating computer system reliability. *Proceedings of the Fifteenth Systems Administration Conference (LISA XV) (USENIX Association: Berkeley, CA)*, page 31, 2001.
- [30] K. Begnum, M. Burgess, T.M. Jonassen, and S. Fagernes. On the stability of service level agreements. *IEEE eTransactions on Network and System Management*, submitted 2005.
- [31] G.R. Grimmett and D.R. Stirzaker. *Probability and random processes (3rd edition)*. Oxford scientific publications, Oxford, 2001.
- [32] J.L. Hellerstein, Y. Diao, S. Parekh, and D.M. Tilbury. *Feedback Control of Computing Systems*. IEEE Press/Wiley Interscience, 2004.
- [33] S. Axsäter. *Logistics of Production and Inventory*, volume 4 of *Handbooks in Operations Research and Management Science*. Elsevier, 1993.
- [34] M. Burgess. *Analytical Network and System Administration — Managing Human-Computer Systems*. J. Wiley & Sons, Chichester, 2004.

- [35] D.A. Menascé and V.A.F. Almeida. *Scaling for E-Business: Technologies, Models, Performance, and Capacity Planning*. Prentice Hall, 2000.
- [36] G. Canright and K. Engø-Monsen. *Handbook of Network and System Administration*, chapter Some Aspects of Network Analysis and Graph Theory. Elsevier, 2007.
- [37] M. Burgess and G. Canright. Scaling behaviour of peer configuration in logically ad hoc networks. *IEEE eTransactions on Network and Service Management*, 1:1, 2004.
- [38] A. Bouch and M.A. Sasse. It ain't what you charge, it's the way that you do it: A user perspective of network qos and pricing. *Proceedings of the VI IFIP/IEEE IM conference on network management*, page 639, 1999.
- [39] J. Topping. *Errors of Observation and their Treatment*. Chapman and Hall, 1972.
- [40] R.H. Dieck. *Measurement and Uncertainty (Methods and Applications)*. Instrument, Systems and Automation Society, third edition edition, 2002.
- [41] M. Sloman. Policy driven management for distributed systems. *Journal of Network and Systems Management*, 2:333, 1994.
- [42] D. Verma et al. Policy based sla management in enterprise networks. In *Policy Workshop 2001*. Springer Verlag, 2001.
- [43] M. Burgess. On the theory of system administration. *Science of Computer Programming*, 49:1, 2003.
- [44] A. Sahai et al. Automated sla monitoring for web services. *IFIP/IEEE 13th International Workshop on Distributed Systems: Operations and Management (DSOM 2002)*, page 28, 2002.
- [45] P. Flegkas et al. Design and implementation of a policy-based resource management architecture. In *Proceedings of the VIII IFIP/IEEE IM conference on network management*, page 215, 2003.
- [46] M. Debusmann and A. Keller. Sla-driven management of distributed systems using the common information model. In *Proceedings of the VIII IFIP/IEEE IM conference on network management*, page 563, 2003.
- [47] Y. Diao, J.L. Hellerstein, and S. Parekh. Optimizing quality of service using fuzzy control. *IFIP/IEEE 13th International Workshop on Distributed Systems: Operations and Management (DSOM 2002)*, page 42, 2002.
- [48] M. Burgess. Computer immunology. *Proceedings of the Twelfth Systems Administration Conference (LISA XII) (USENIX Association: Berkeley, CA)*, page 283, 1998.
- [49] M. Burgess. Two dimensional time-series for anomaly detection and regulation in adaptive systems. *IFIP/IEEE 13th International Workshop on Distributed Systems: Operations and Management (DSOM 2002)*, LNCS 2506:169, 2002.
- [50] Y. Diao et al. Generic on-line discovery of quantitative models for service level management. In *Proceedings of the VIII IFIP/IEEE IM conference on network management*, page 158, 2003.
- [51] Arun V. Holden, editor. *Chaos*. Manchester University Press, 1986.
- [52] H.L. Lee and S. Nahmias. *Logistics of Production and Inventory*, volume 4 of *Handbooks in Operations Research and Management Science*, chapter Single Product, Single Location Models. Elsevier, 1993.

- [53] M. Burgess and T. Reitan. A risk analysis of disk backup or repository maintenance. *Science of Computer Programming*, (to appear), 2005.
- [54] M. Burgess. Probabilistic anomaly detection in distributed computer networks. *Science of Computer Programming*, page (To appear), 2005.

Service Provisioning: Challenges, Process Alignment and Tool Support

Michael Brenner¹, Gabi Dreo Rodosek², Andreas Hanemann³,
Heinz-Gerd Hegering³, Ralf Koenig¹ *

Munich Network Management Team, Germany

{brenner, dreo, hanemann, hegering, koenig}@mnm-team.org

November 28, 2006

^{*1}Ludwig Maximilians University Munich, ²University of Federal Armed Forces Munich, ³Leibniz Supercomputing Center Munich

Contents

1	Introduction	4
1.1	Terms	4
1.2	Differences in Provisioning due to the Varying Level of Customization	6
1.2.1	Provisioning of Commodity Services	8
1.2.2	Provisioning of Individual Services	10
1.2.3	Provisioning of Customized Services	12
1.3	General Phases of Service Provisioning	12
1.3.1	Service Planning	12
1.3.2	Service Ordering	13
1.3.3	Service Fulfillment (Deployment and Configuration)	13
1.3.4	Service Operations	14
2	Service Provider Goals and Challenges	14
	Goal 1: Achievement of business objectives and high service levels	14
	Goal 2: Involve customer in service ordering	15
	Goal 3: Automation of service fulfillment	16
	Goal 4: Optimal resource selection and utilization	17
	Goal 5: Preparation for automated service operations	18
	Goal 6: Adaptivity to changes caused by technology, customers or market demand	19
3	Process Frameworks and Concepts	19
3.1	Enhanced Telecom Operations Map (eTOM)	20
3.1.1	Introduction	20
3.1.2	Structure and Content	20
3.1.3	Service Provisioning in eTOM	23
3.1.4	Summary	23
3.2	IT Infrastructure Library	24
3.2.1	Introduction	24
3.2.2	Structure and Content	25
3.2.3	Service Provisioning in ITIL	25
3.2.4	Summary	30
3.3	Service Oriented Architectures	30
4	Trends in Technologies and Tools	31
4.1	Service Planning	32
4.2	Service Ordering	32
4.3	Service Fulfillment	32
4.3.1	Resource Virtualization	32
4.3.2	Tool Support for Resource Configuration	33
4.3.3	Web Service and Grid Service Standards	33
4.4	Service Operations	35
4.5	Service Provisioning Workflow Support	35
4.6	Summary	36

5	Research Activities	36
5.1	Service Planning	36
5.2	Service Ordering	36
5.3	Service Fulfillment and Operations	37
5.4	Information modeling	39
6	Service Provisioning in Practice and Operations	40
6.1	Best Practices in Resource Allocation and Assignment	40
6.1.1	Late assignment of resources to requests	40
6.1.2	Soft-state assignment of resources to requests	40
6.1.3	Configurable resources with remote management support	41
6.1.4	Resource virtualization	41
6.1.5	Overprovisioning	42
6.1.6	Oversubscription	43
6.2	Comments by Practitioners in Service Provisioning	43
6.3	Selected Scenarios from Different Articles (Case Studies)	45
7	Summary	46

1 Introduction

The provisioning of services in the field of information and communication technology (ICT) such as network and application services presents many challenges to service providers. Service providers must pay careful attention to operational efficiency and take new approaches to service provisioning in order to compete successfully. They must make better use of existing resource infrastructures, such as networks and servers, as well as reduce the number of operational staff required to deliver services to end users leveraging automation.

Regarding the level of service customization, services span a broad range: from widely deployed commodity services, such as broadband internet access or e-mail services, with millions of equal service instances, to individual services, such as an enterprise extranet or an in-house service like the service architecture of a world wide web (WWW) search engine with a single highly customized service instance.

The vision of a service provider and also a competitive advantage is the achievement of “flow-through provisioning”, which means that a customer reviews a set of services offered to him by a provider, selects an appropriate service, chooses among the available service quality options, then places an accordingly formed service order, and waits for its almost instantaneous fulfillment. The service provider, who receives such an order, would have automated provisioning systems that activate, monitor and manage the ordered services with minimal additional manual input. So far, this has partly been accomplished for a few types of commodity services, but it remains a vision for most services that require a higher level of customization.

1.1 Terms

We want to motivate service provisioning and most of the terms used in this chapter with a short example of a commonly known commodity service subscribed by many customers.

Example: A customer who wants to order broadband cable access to the Internet typically does so by submitting an order form. He fills in the blanks and chooses among the several available options associated with service parameter values, such as link bandwidth and monthly traffic volume. Finally, after selecting a payment option, he sends the form to the service provider. By executing these actions, a customer has ordered a service.

The form is processed by the service provider and data about the customer and the requested services is registered in some data store. As broadband cable is a shared medium, the internet service provider (ISP) evaluates whether this additional customer can be attached to the particular cable segment that links his home to the cable network, typically based on some (non-disclosed) over-subscription model. If there are too many customers subscribed to a particular segment to keep up with performance guarantees at peak times, the cable operator may have to split the segment. After updating systems for authentication, authorization and accounting, the corresponding network components are configured accordingly to allow traffic from the new customer through the service provider’s network to the Internet and vice-versa.

Eventually, a technician is sent to the customer to install and test the interface at which the customer can get access to the ordered service. Depending on the

contract, a pre-configured cable modem may be ordered by the service provider and shipped to the customer.

Speaking in more general terms, a *service* is a concept how a certain *functionality* is provided to a *customer* by a *provider*. It abstracts from the specific implementation while all customer-relevant service parameters (e.g., Quality of Service (QoS), cost parameters), such as agreed upon bandwidth or availability in network services, are specified in a *Service Level Agreement (SLA)*.

The service provider is usually free to change the implementation of the service as long as the SLA is not breached. With such an approach, the growing number of shared and dedicated *resources* needed to provide a service as well as the details about the processes needed for its implementation and operation are wrapped for the customer behind a more pleasant interface. This allows the service provider quite a degree of flexibility in the design and management of its resource infrastructure and enables it to leverage various economies of scale and scope, see [1] for details.

The separation of functionality (customer view) and implementation (provider view) is less applicable when customers want specific services being planned and built with regard to their individual requirements, such as existing legacy systems and services that the ordered service has to work together with, thereby limiting the range of implementations for the service provider.

In order to analyze service provisioning in more detail it is necessary to start with the *service life-cycle*. IT services can be associated with a service life-cycle that subsumes the steps from planning to termination of a particular service. A popular simple service life-cycle is called *Plan-Build-Run*, typically re-cycled after a *Change* or *Improvement* step. *Service provisioning* mainly deals with the plan, build, and change parts, providing the necessary input for run-time operations. It is therefore a major part of the service life-cycle. More precisely, service provisioning includes tasks such as planning new services, building the basic infrastructure, SLA negotiation and order processing, identifying adequate resources for service delivery or adapting existing services to specific customer's needs, specifying steps for service implementation and service operation, up to dynamic, near real-time service composition out of service modules based on customer requirements.

In order to offer services at competitive prices, a service provider must work *efficiently* by using as few resources as possible in terms of staff and machinery, while providing services *effectively* by meeting or exceeding the agreed SLA. Therefore, resources are typically shared between multiple customers as long as the SLA does not bind the service provider to use dedicated resources. With new virtualization technologies appearing on the market, the border between physically separated resources and logically shared resources becomes increasingly fuzzy.

Example: Imagine the case that five customers need a dedicated server environment with privileged access but they do not need the full computing power of a physical server due to a particular workload profile. To fulfill these service orders, one possible implementation is to configure several virtual servers on one physical server, based on the assumption that the workload profiles overlap only in short periods of time. This way, the service provider can use virtual servers to provide abstract dedicated servers, saving four physical servers including expenses for energy, cooling and rack space. While the service provided to the customer is essentially the same, the implementation is different.

Examples for other shared resources are network links and components shared by multiple virtual private networks or virtual LANs, file servers shared by multiple users, or virtual hosts on a HTTP server. Thus, resource sharing is an optimization method of a provider with regard to resource utilization. Sophisticated mechanisms and tools are necessary to support such sharing to assure that guaranteed service levels are not violated.

When providing *best-effort services*, service providers do not make any service guarantees. The low complexity in service management saves efforts on the provider side and results in lower prices. Best-effort is the typical service quality found in the public internet.

Advanced *quality of service* mechanisms, e.g. reserving a certain portion of end-to-end network bandwidth on demand for video traffic using the Resource Reservation Protocol (RSVP), need to be implemented to provide guaranteed service quality. Many steps in the service provisioning process deal with the assurance of a specified service parameters.

1.2 Differences in Provisioning due to the Varying Level of Customization

Services can be classified regarding a number of properties. For service provisioning, relevant aspects are mainly the number of customers that will subscribe to the service, and the level of customization that these customers require. The number of service subscriptions directly translates to the number of times the order process and the following steps will be executed, while the level of customization translates to the complexity of these processes and therefore the applicability to automated execution.

Regarding the level of customization of service instances, IT services can be roughly grouped into the following three types of services: commodity services, customized services, and individual services. Please note, that the line between these service types is not always sharp and some services may show properties from multiple categories. In the next sections, we will first describe the extremes of this spectrum, namely commodity and individual services, followed by a description of customized services filling the wide range between the former two.

Table 1: Types of IT services with respect to the level of customization

	Commodity services	Customized services	Individual services
Requirements on the customer	customer has to meet certain preconditions	depends on service offer, requirements may be up to negotiations	None, customer requirements are evaluated and implemented
Service parameters	are widely standardized across service providers, customer has few service configuration options	customer has multiple options to choose from a predefined set of service parameters	the set of service parameters is up to negotiation and can change at run-time
Service orders	sold/subscribed at high volumes of equal orders, low price per subscription	low number of equal orders, but large number of orders	one order with substantial budget
Pricing	high cost pressure for service providers as services are easy to compare for customers	higher prices may be justified by the level of customization which provides substantial value to the customer	typically expensive to plan and build, individual pricing, customization can pay off in service usage
Implementation	hidden from customer, service provider is free to change implementation without prior notice to the customer	typically made by combining and integrating service modules with customer-specific glue code, customer can influence implementation within certain bounds	customer requirements can have substantial influence on implementation, reusability of the result is not a key issue
Provisioning tool support	good tool support for provisioning, a high degree of automation	applicability of tools depends on a number of issues (see text)	few tools available, most work done by experienced solutions designers
Support and Maintenance	costly on-demand support, when not using the least cost intensive way of communication, support is regarded as a cost driver	depends on service offer, service provider may serve as mediator between resource suppliers and customer	typically sold with support and maintenance contracts, support is regarded as a source of income for the provider

1.2.1 Provisioning of Commodity Services

Commodity services are affordable, widely available, well-specified standard services with known functionality provided by many competing service providers. Examples include voice services, broadband internet access, or web hosting.

See Figure 1 for a schematic overview on the provisioning of commodity services. The introduction of new commodity services is typically initiated by service providers sensing market demand and not by specific customer requests.

Commodity services require a basic infrastructure to run services upon, which at first needs to be planned, built, rolled out, and set up by the service provider. Resources are added later on based on market demand analysis and predictive planning.

Resources in this infrastructure and its management are planned and laid out in such a way that service orders from customers can be processed and provisioned in a highly automated fashion, given the fact that the ordering and fulfillment process is run possibly a few thousand or million times. Operations support systems (OSS) and management systems control as many steps as possible in the provisioning workflow, but they need orchestration of devices as well as up-to-date information about the deployed infrastructure.

Commodity services are intended to be subscribed by a large number of customers with a low level of customization, so services are typically comparable even across service providers, mainly for interoperability reasons.

Prices and service parameters are determined by the service provider for pre-packaged service bundles with some service options and are hardly subject to negotiations. Therefore, customers usually have to accept the conditions stated by the service provider and can only search for a service offer that best fits their needs in terms of service features and price.

A large number of essentially equal service orders allows and even requires a high level of automation in the ordering and fulfillment process from service providers in order to successfully compete in the market.

With commodity services, the implementation of a service is typically hidden from the customer, with the service provider being free to change service implementation without prior notice. While service planning and setup of a service infrastructure can take months or years, the service ordering and fulfillment phases for a single service order can be completed within a time range of seconds to days, made possible by the high level of automation.

Due to the high volume of resources in such infrastructures, suppliers offer hardware and software tailored to the requirements of commodity service providers with many configuration options.

The basic infrastructure for many commodity *network* services has been established by state-owned companies in many countries of the world. Market liberation and regulation efforts require established network service providers to open their infrastructures or to provide transit services at certain rates. This led to the establishment of virtual service providers and resellers that use the existing operating infrastructure.

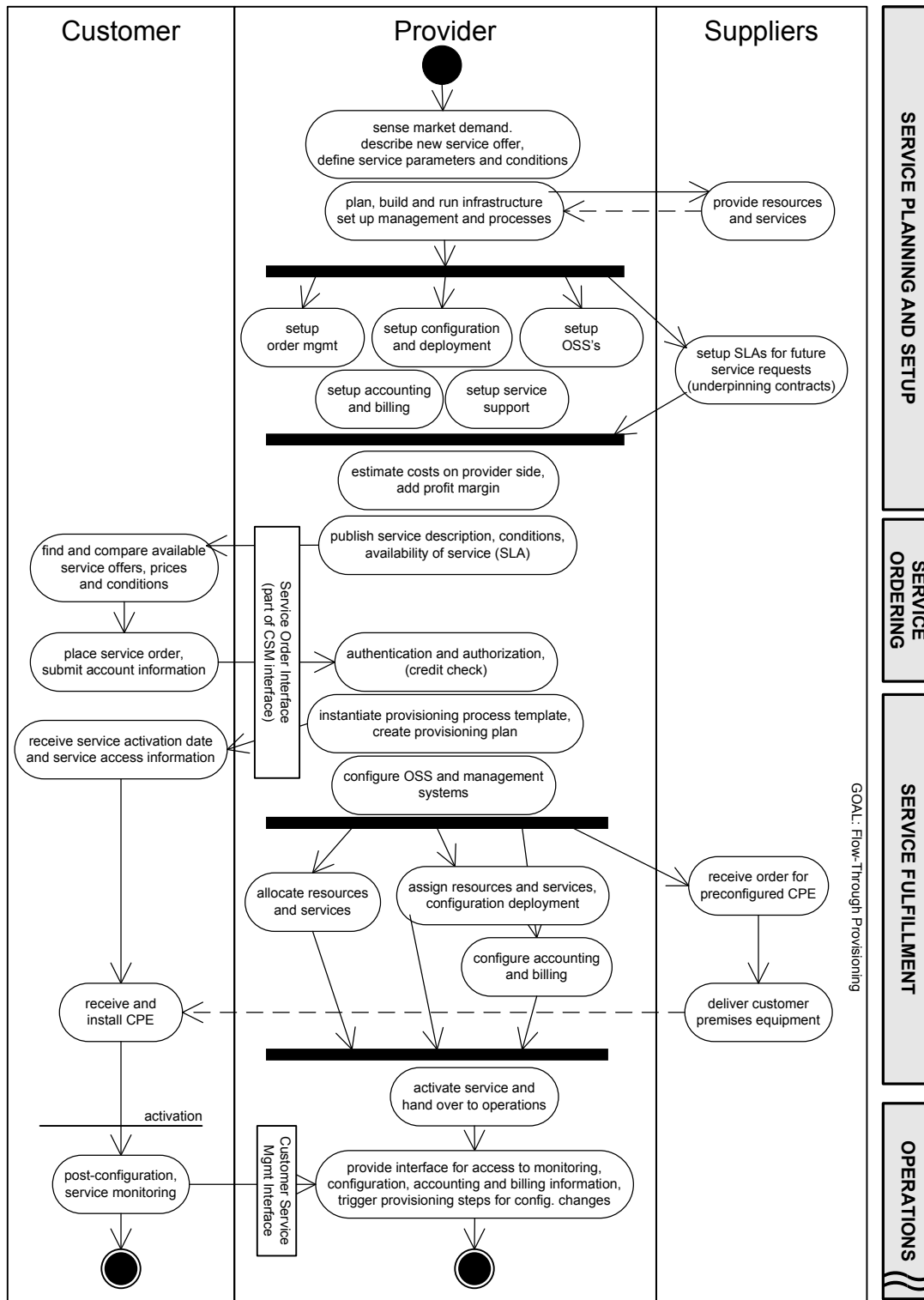


Figure 1: Provisioning steps for commodity services

1.2.2 Provisioning of Individual Services

Since *individual services* or *solutions* are sold to a single customer, individual negotiations of service parameters and values for service level objectives are an appropriate way to find a compromise between costs and customer requirements.

Individual services for major enterprise customers are typically organized as long-term projects. This is caused by the high level of required customization, resulting from customer-specific requirements documents of substantial size.

See Figure 2 for a simplified, schematic overview on provisioning of individual services.

The provisioning of individual services starts with the customer compiling a service request with detailed specifications of the customer requirements, usually expressed in terms at business process or service level. Many customers use consulting services in compiling this document.

Multiple service providers analyze the service request and refine customer requirements and SLA individually together with the customer. Each provider identifies a set of appropriate resources and subservices to compose or build the solution. After estimating costs and adding a profit margin, bids are sent to the customer. At this point, in most cases there is still a considerable level of uncertainty due to changing or non-specified customer requirements, estimates, and safety margins on both sides.

When the customer decides to accept an offer and places a service order, both sides agree on service functionality, service levels and costs. However, customers may demand specific interfaces, extensions or capabilities not commonly offered by suppliers. In addition, often existing resources or services of the customer may have to be integrated with the new ordered service. Automation in provisioning of such requests is therefore hard to achieve.

Service provisioning for individual services is more project-driven, which involves many people communicating and negotiating with each other. Tools that support the collaboration of people such as project management tools, a knowledge base for past solutions and group communication tools provide valuable help and guidance.

Provisioning time – this includes service planning, ordering and fulfillment – is in the range of months to years. Key issues are the fast and thorough analysis of customer requirements and mapping them to needed resources, as well as good communication skills. Experience in building individual services from flexible standard modules by service composition can substantially reduce new developments and therefore provisioning time.

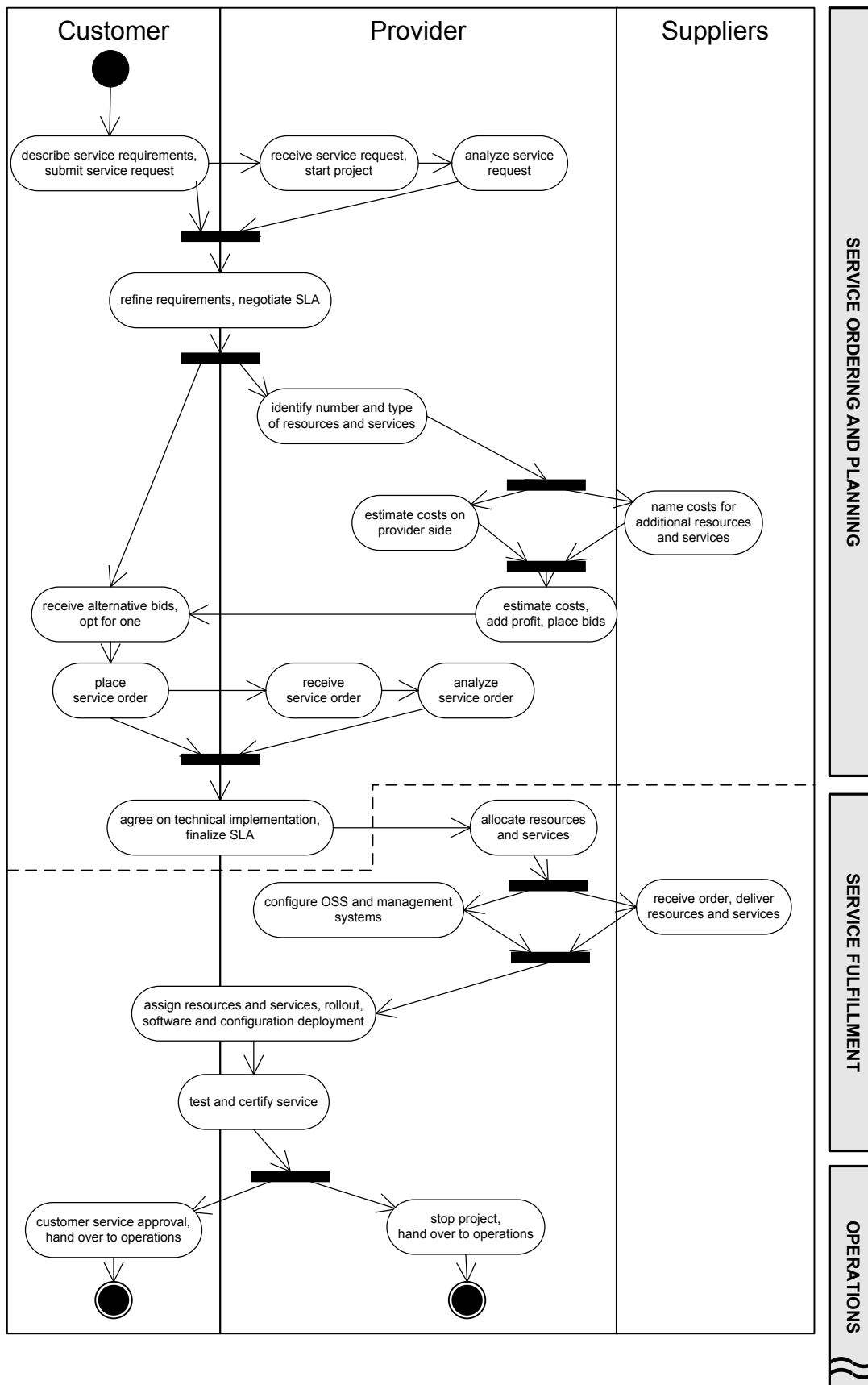


Figure 2: Provisioning steps for individual services

1.2.3 Provisioning of Customized Services

Customized services fill the wide range between the former two types of services.

In many cases, customized services are developed by adding more configuration and service options to existing commodity services to better meet the anticipated requirements of a potential set of customers. Other providers develop customized services from individual services. By refactoring the composition of the service into service modules and separating configuration options from service implementation, the former individual service becomes more flexible so that the requirements of more than one customer can be met and development costs per customer decrease.

Customized services typically have a substantial number of orders, but due to the large number of service options and choices for the customer, hardly any two service orders are exactly the same. Depending on the specific service, customized services show features of both commodity and individual services. Customized services are typically built from standard service modules, which allows semi-automatic provisioning. A certain degree of flexibility for the customer is made possible by having the choice between different types of modules and a custom arrangement of the modules to a customized service. This results in a substantial growth of feature combinations with numerous dependencies. Automation would require to model all this complexity, which in turn requires models for all resources and subservices, and thus limits flexibility. The most appropriate level of automation is therefore determined on a case-by-case basis based on many criteria such as cost of human labor, cost and flexibility of automation tools, uniformity of service orders, existing knowledge, supplier relationships and negotiation skills to name just a few.

For comparison: In manufacturing, products built in a similar fashion are called to be “built to order”. To compete in the market, the sum of costs for development, customization, resource consumption at runtime, and management of customized services must be optimized, which requires advanced skills in supply chain management of a number of specialized suppliers.

1.3 General Phases of Service Provisioning

As already indicated in the previous discussion, service provisioning consists of the phases (i) service planning, (ii) service ordering, and (iii) service fulfillment (deployment and configuration). Once these steps are completed, the service is activated and handed over to service operations.

The phases were visualized in Figures 1 and 2 for commodity services and individual services. In contrast to the simplified schematics in the two figures, provisioning of all three types of services can be a joint effort of many actors with different roles. Individual services can be planned and built by solutions providers, while service operations is handed to the customer or a third-party service provider. In customized services, service integrators may take over specific sub-tasks to integrate a certain set of resources. Resource suppliers may also offer services to ship resources pre-configured to service provider requirements.

1.3.1 Service Planning

In this phase a service provider plans the provisioning of a new service. Planning includes several steps ranging from a market analysis to determine the business model, the appropriate costs and charges to the verification of technical issues such as the availability of resources. In case of commodity services, that are provided to a large number of customers, a detailed market

analysis is a prerequisite for successful service introduction. The trigger to set up and provide such services results mainly from decisions on the provider side. In contrast, if a new service is requested by a customer, which is the typical case with individual services, the planning process is limited to the verification of the available resources and other provider-specific issues, such as policies.

A more detailed observation identifies the following steps:

- Definition of the service and its functionality, service parameters and possible service levels that can be offered to customers
- Definition of a cost model for internal and external costs
- Sizing: quantitative estimation of market demand for the service, quantitative mapping of market demand to service demand, quantitative mapping of service demand to resource demand
- Verification of the available infrastructure (resources) as well as capacity planning with respect to the number of customers and provided service levels
- Planning processes for resource allocation, resource assignment, service deployment (roll-out), and initial configuration
- Specification of a service ordering process, definition of allowed service changes
- Planning service operations including service management tools

1.3.2 Service Ordering

During the service ordering phase, a customer selects a service or a service package from the available list of services that a provider offers to customers. By selecting a service, a customer identifies not only the functionality of the service itself but also the requested service quantity and quality, the service cost, the time line when he wants to use the service. The specification of these parameters is done with respect to the agreed values and value ranges as well as thresholds in the respective SLAs. Values, value ranges as well as other issues can be negotiated between a customer and a provider.

1.3.3 Service Fulfillment (Deployment and Configuration)

The service fulfillment phase starts in fact after the submission of the service order. The service provider needs first to check the order with respect to contractual and technical issues and either rejects or grants the service order.

If the service order can not be granted, it is also possible that the customer and the service provider negotiate about the requested service. If a provider grants the service order, he starts the service fulfillment process, namely resource deployment and configuration as well as the initial configuration of management systems.

Finally, the service provider informs the customer about the successful realization of the service order, and the customer may start to use the service. On the provider side, the service is handed over to the operations department.

1.3.4 Service Operations

After service activation, service operations deals with the day-to-day operations to provide services with constant quality meeting or exceeding the SLA.

As a matter of fact, hardware and software resources fail from time to time for a number of reasons and human operators make mistakes. To assure to be notified in case of service failures and degradations of service quality, monitoring the infrastructure is considered necessary by service managers.

Since services are provided upon resources, resource and service management tools must be configured appropriately to gain a service-oriented view that provides the “right” amount of service-level information to managers from existing resource-level monitoring data. In turn, service management operations must be mapped to operations at resource level.

In addition to providing information to service managers, as well as system and network administrators on the provider side, operations should provide customer-relevant management information via the Customer Service Management (CSM) interface.

2 Service Provider Goals and Challenges

Awareness of the main goals of service provisioning is a prerequisite for identifying fundamental challenges of service provisioning that a service provider is confronted with.

While the ultimate goal for a service provider is economic prosperity and providing agreed service levels to his customers, in service provisioning this translates to tuning many parameters to find the right balance between multiple sometimes contradictory goals. Each goal is associated with challenges that have to be faced in terms of meeting the goal. The following discussion will outline the relationship between goals and challenges. It should be noted that the list is not exhaustive although the aim was to provide a broad view; it should give just an idea about the problem area we are talking about.

Goal 1: Achievement of business objectives and high service levels

In today’s business environment, services are rolled out in almost real-time. The challenge is of balancing cost-effectiveness with resource utilization needed to meet the dual requirements of ensuring high availability and maximizing SLA performance. Today, “available” means the right response time, to the right users, right now. So, slow performance, delayed reaction to customer service requests etc. is no longer an option. In fact, the essential goal of a service provider is to move from bits to business values which is also reflected in the goals that a provider has to achieve. However, it differs when talking about individual or commodity services. For commodity services a provider starts a large market analysis to see what the market requests whereas the provision of an individual service is triggered by a specific customer. Business value is in the latter calculated customer-specific.

Challenge 1.1: How to formalize a SLA? So far, no generally agreed upon formalized SLAs exist yet. Rather, the inclusion of elements into a SLA (e.g. specification of a service, the service access point (SAP), QoS parameters, cost parameters, accounting models, security levels) depends on a case-by-base basis.

Agreeing on SLAs without a deep knowledge on the area can lower business objectives in a great extent. For example, if specifying only the QoS parameter availability without specifying

also the maximal time of an outage, this can lead to a situation where the availability is met, however, the customer is not very satisfied. Beside the challenge to identify all relevant elements of a SLA from the customer and provider side, the resulting automatic service and resource configuration based on SLA parameters is another challenge.

Challenge 1.2: What service levels can be offered, resp. are required by customers? This challenge depends on the type of service. In case of commodity services a detailed forecasting and what-if analysis are a precondition to determine adequate service levels that can be offered by a provider. Service levels for individual services are determined on a case-by-case basis. The risk to determine service levels that do not meet business objectives is minimized.

Challenge 1.3: How can the negotiation process between a customer and a provider be supported? This challenge refers to the support of the negotiation process between a provider and a customer of the possible service levels. It includes also the contractual and technical verification of customer service requests. Afterwards, the service configuration is started. But service configuration is becoming more difficult all the time, thanks to challenges like personalized service, hybrid networks and end-to-end service guarantees, sending fulfillment costs spiraling.

A negotiation about the service levels between a customer and a provider as well as suppliers can include several iterations until an acceptable solution with respect to cost, quality and other aspects is found.

Before negotiation is started, the verification of contractual and technical issues is performed. The verification of contractual aspects is more or less straightforward. However, the verification of technical issues refers mainly to the resource allocation problem, including the identification of the adequate resources, verification of their availability etc. It is necessary to verify whether other services, respectively customers, may be affected in the provided quality of service if the new service request is granted.

Goal 2: Involve customer in service ordering

As more and more steps in service provisioning are automated with the help of IT systems, all input data must be converted to an electronic form-based format, interpretable by machines.

Traditional points of contact between customers and providers have been branch offices and service hotlines. There, customers expressed service orders or desired changes to existing services, which were then translated by support staff to entries into an internal ordering system. In the same way, output data like service usage statistics and bills, while being generated electronically, was printed and mailed to the customer when internet access had not been widely available to customers.

Disadvantages of this practice on the provider side include expenses for branch offices and staff, printing and mailing. On the customer side, customers could hardly see the full list of available service options and consequences. In addition, call times and opening hours of branch offices are typically limited to about 8 hours per day.

Challenge 2.1: How to involve the customer in the service provisioning process? To involve the customer in the service ordering and service change process has two benefits. First, a customer is more satisfied if he is informed about the status of the service provisioning process and he can dynamically change his service orders within the limits of the SLA. Second,

by shifting the service ordering and service change process to the customer, a provider can concentrate on service provisioning and can save resources and staff, that would otherwise be needed for receiving service orders and change requests.

A precondition to achieve this goal, however, is to provide a suitable Customer Service Management (CSM) interface where service parameters can be changed or new services can be ordered in a convenient way by the customer in a self-service manner.

Challenge 2.2: How to design an appropriate interface to the customer that supports all interactions between a service provider and a customer (e.g. service ordering, reporting of trouble reports, accessing performance statistics about the provided service quality etc.)?

A customer may order services or even service packages. In case of the negotiation process it is necessary to think of various types of services as identified already in the introduction. In case of commodity services such a negotiation is almost impossible, in case of individual services it is almost a certainty.

Online customer service management portals, which are typically web-based, more and more replace traditional customer service management interfaces. Via the CSM portal, the service provider receives customer service requests in a known and machine-readable format and provides machine-readable output data like bills and usage statistics, which on the customer side can serve as input for IT systems for electronic payment or accounting.

Goal 3: Automation of service fulfillment

The service provisioning process is quite challenging since the provider has to deal with distribution of services and the inherent heterogeneity of the infrastructure. The heterogeneity arises from the application of many resources from different vendors (technical heterogeneity), the involvement of a variety of different parties within and outside the service provider side (intra-organizational and inter-organizational heterogeneity). Beside of this static complexity, the type and number of changes lead to a dynamic complexity of the service provisioning process.

Depending on conditions like the level of customization of the service (commodity service, customized service, individual service, see Section 1.2 for details) and the organizational structure of the service provider, standard provisioning workflows should be defined for each offered service. Each workflow specifies the collaboration among all involved people and systems. While this often implies features of bureaucracy which appears to limit flexibility, it reduces coordination efforts and works towards a uniform, measurable performance of service provisioning. To achieve higher levels of automation, the provisioning workflow, including steps, roles, and policies must be expressed in machine-interpretable terms.

Challenge 3.1: What are the necessary service fulfillment and service operational steps? How can these steps be acquired, maintained and improved? How can the acquisition and the execution of these steps be supported by tools or even automated? In order to realize “flow-through provisioning” it is necessary to automate service provisioning as far as possible. This means to formalize the necessary provisioning steps and refine them into commands executable by machines, with parameters so that the workflow can be instantiated for similar deployment tasks. These workflow descriptions are collected in a repository of *service provisioning templates* internally within the provider’s environment or even externally so that several providers can access it.

Another problem to address is the alignment of service fulfillment steps with the current infrastructure, which refers to the ability to change the steps with respect to changes in the environment either on the service or resource layer. Such an alignment should be done “almost automatically” by management tools with minimal necessary assistance from operators. In case of an individual service such steps are probably service-dependent and need to be done manually. However, in case of a commodity service such service provisioning steps need to be predefined and executed in an automated way.

The problems associated with the specification of deployment and configuration steps are:

- the granularity of the steps,
- the workflow aspect and
- the alignment of service provisioning to the changing set of resources.

Granularity addresses the refinement of steps to a level of directly executable actions, which ultimately can be performed automatically by existing management tools. An abstract action such as *install_database* needs to be mapped to a more concrete step like *install(oracle9, server1)*, which in turn must be refined to include more details about installation parameters. Given the number of hardware and software resources, resource and service dependencies, distribution of knowledge and the rate of change in all mentioned entities, this presents quite a challenge for complex service provisioning scenarios.

There are two approaches to define provisioning and operational steps:

- to specify all steps with the requested granularity in the initialization phase or
- to specify only a limited set of steps in the initialization phase and improve them afterwards through experience gained during the application of these steps.

The first approach is applicable for environment with simple services and small infrastructure. In case of complex services that depend on other services and large heterogeneous infrastructure the second approach is probably more applicable.

Challenge 3.2: How fast can a new service be deployed? Customers may request the provision of new services or changes of the already provisioned ones. In both cases it is necessary to identify the (i) involved resources in terms of resource allocation, (ii) the management tools that are involved in service operation and (iii) the processes that need to be eventually adapted or defined in addition. A precondition for a fast provisioning is an extensive and integrated usage of management tools in order to have the necessary management information about the infrastructure available. Supporting demanding service provisioning means always also to be able to forecast the service usage with respect to e.g. subscribed SLAs, provider policies, measurements and historical data. Furthermore, sophisticated resource planning methods are needed to accomplish this goal.

Goal 4: Optimal resource selection and utilization

While flow-through service provisioning primarily refers to automation of service provisioning processes, optimal resource utilization deals with the question of how the existing infrastructure (i.e. machines and staff) can be used in an optimal way. Talking about the “optimal” solution

raises immediately the question about the objective function of the optimization e.g. whether it is necessary to optimize with respect to service quality, availability, security aspects, costs, specific customer demands or other provider-specific policies. Besides the infrastructure aspect, optimization refers also to the service provisioning process itself. Efficient process definition is quantified with so-called key performance indicators (KPIs).

Challenge 4.1: How to describe service requests and resource capabilities in an adequate way in order to optimize resource allocation with respect to various provider objectives?

This challenge refers to mapping *service requests* to *resource capabilities*. In case of a formalized description of service requests and resource capabilities the mapping is more or less straightforward. However, in addition several provider policies, high availability requirements from customers can influence service provisioning as well, and make the resource allocation problem a hard problem.

Challenge 4.2: Since several solutions of the service-resource allocation problem are possible, how to support a decision process to select the “right” one? How can such a decision be supported by management tools?

If several solutions for a resource allocation are possible, it is necessary somehow to determine the “right” one for the special case. Thus, it is necessary to introduce metrics to evaluate what solution is the most appropriate. In this context, what-if analyses and simulations are very appropriate approaches since a provider can change the values of input parameters and evaluate the influence on the result.

Resource selection becomes even more challenging when considering that resources and sub-services can also be bought from suppliers. As provisioning deals with resources and services, how to decide whether a certain service or subservice should be hosted by the service provider himself or ordered from a specialized supplier? From which supplier to buy a particular resource resp. subservice?

Challenge 4.3: How to assure that a new resource to service assignment will not affect the provided quality of service of other customers or violate provider policies? How to support such what-if analyses?

What-if analyses support the decision process whether to deploy new services upon the existing infrastructure or to extend existing ones or even change the infrastructure of policies. Managers can easily see where demand exceeds supply, where skills need to be enhanced or developed, or if new staff members need to be brought in.

Goal 5: Preparation for automated service operations

A precondition for service operations is to obtain a service-oriented view of the infrastructure which refers to the issue how to configure device-oriented management tools appropriately to gain a service view. This means also to keep track of the relationships and dependencies between services in terms of inter-service dependencies as well as the dependencies between services and resources. Key issues here refer to the (automatic) acquisition of dependencies and the offering of this information to processes, tools and staff that needs this information for various purposes (sales, operations and monitoring, accounting). Furthermore, tracking the distributed provisioned services upon resources is another important aspect as well.

Challenge 5.1: How to specify the service monitoring process?

This challenge refers to the mapping of service levels to the design of the service monitoring process. If it is for

example agreed on resolve times below a few hours, this is necessary to be supported by several processes, for example incident and problem management process. The efficiency of the processes is measured with key performance indicators as described latter on.

Challenge 5.2: How to configure the device-oriented management tools to monitor the behavior of resources involved in the service provisioning? Beside the negotiation process itself, operational requirements for various types of services differ as well. Individual services can be handled manually whereas for services offered to hundreds of customers only an automatic approach is a reasonable solution. Furthermore, services can be provided by several providers. In such a case it is necessary to address issues such as what happens if a sub-provider violates the SLA with the provider, and what are than the consequences for the SLAs between the customer and provider.

This challenge deals with the mapping of QoS parameters as specified in a SLA (e.g. availability) to management variables such as `node_up` or `node_down`.

Goal 6: Adaptivity to changes caused by technology, customers or market demand

An essential competitive factor among service providers is the ability to realize customer service requests in a fast and convenient way to keep existing customers resp. gain new ones.

Challenge 6.1: How fast can changes be realized? Changes can result due to customer requests, as a result of a planning solution or due to changes of provider's policies. Since changes in the infrastructure result in most cases also into changes in the service fulfillment, it is necessary to adapt to these changes in a fast and convenient way. This goal addresses the provider's view of the dynamics of provisioning. It is an important goal as well as a big challenge for a service provider to cope with the dynamics of changes, either of the provided services, the used infrastructure or provider policies. Furthermore, the dynamic nature of SLAs, service and resource configurations, and resource usage needs to be addressed as well.

The ability to adapt to changes means to store service and resources models and keep an up-to-date inventory of all provisioning-related information and policies, and their relationships. Certainly, a periodical verification and update of models as well as inventories, including changes in the infrastructure, is essential.

Challenge 6.2: How to perform provisioning in a scalable, reliable way, regarding to service/subscription types and order volume? To ensure reliable provisioning with few errors, especially in the implementation of automated procedures, means to refer to the point of high availability and reliability of the provisioning system, "carrier-class" as well as to reduce risk in provisioning errors.

Due to the distributed nature of service provisioning, in most cases several persons are involved in the service provisioning process. Their actions need to be specified and coordinated in an appropriate way which brings us to e.g. ITIL, eTOM, as described in the next section.

3 Process Frameworks and Concepts

This section will outline two current concepts of increasing relevance for IT service provisioning: organizational IT Service Management, represented by the two IT business process frameworks

ITIL (IT Infrastructure Library) and eTOM (enhanced Telecom Operations Map), and SOA (Service Oriented Architectures), a new paradigm for building service infrastructures. ITIL and eTOM, discussed in the following two sections, present relatively mature frameworks – TOM, the predecessor of eTOM, was first published in 1998, the earliest ITIL publications date back to the late 1980’s. SOA by contrast is a currently still evolving idea, that has yet to result in significant standardizations, and will therefore be discussed only briefly in the last section.

3.1 Enhanced Telecom Operations Map (eTOM)

The eTOM [2] is a process framework that started as an effort for telecommunications provider process standardization, but – influenced by increasing convergence of telecommunications and classic IT services – has evolved into a comprehensive IT service management standard.

3.1.1 Introduction

The “e-commerce” or *enhanced Telecom Operations Map* (eTOM) is a business process framework for *Internet and Communications Service Providers* (ICSP). It is owned, published and maintained by the TeleManagement Forum (TM-Forum or TMF, formerly known as the NMF or Network Management Forum). Its predecessor, the Telecom Operations Map (TOM) was first published by the TMF in 1998 and was superseded by the eTOM in 2001. The goal of TOM was the creation of an industry-owned framework of business processes, including the definition of a common enterprise-independent terminology for service management. It was also supposed to serve as a basis for discussing the scope of management information necessary for the execution of the processes. The results of the latter effort have meanwhile spawned their own TMF document family, the *Shared Information and Data Model* (SID) [3]. Together with the guidelines on *Technology Neutral Architecture* (TNA) [4], a lifecycle model [5] and the development of conformity tests [6], eTOM and SID form the pillars of the ambitious *New Generation Operation System and Software* (NGOSS) project [7]. NGOSS aims to define a vendor-independent architecture for management systems, which will permit the construction of complete management solutions from modules with standardized interfaces. Since 2004, eTOM is also an ITU-T Recommendation (M.3050).

3.1.2 Structure and Content

After a brief introduction to the organization of the eTOM document family and the basic structuring principles underlying the herein documented framework, the general content and structure of the *Operations* process area, which is the most relevant to service provisioning, is presented.

Documents Even though the eTOM makes up only a part of the NGOSS project, its description consists of several documents. Various important additions to the main framework document [2] are made in the addenda and application notes. The application note C [8] is a rough draft of a “Public B2B Business Operations Map”, addendum D [9] contains “Process Decompositions and Descriptions”, addendum F [10] “Process Flow Examples”, and finally the “eTOM ITIL Application Note” [11] is a brief outline of the relationship between ITIL and eTOM processes. While the application notes are useful, they are not as essential as the addenda, which in parts contain information critical for the use of eTOM.

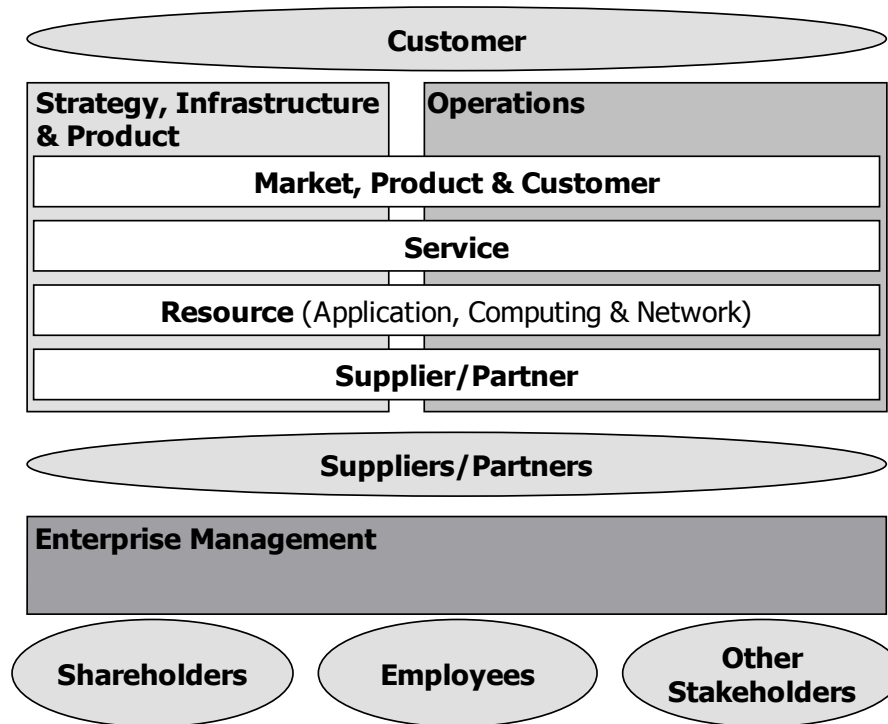


Figure 3: eTOM Level 0 View [2]

Framework Structure The eTOM features different views on its processes, defining five *view levels*, from level 0 to level 4. Numerically higher levels offer an increasingly finer view of the service provider processes. In level 0, only three fundamental processes (in level 0 also called “process areas”) are defined, while on level 2, the only level for which process definitions for all level 0 process areas exist, there are already 72 processes. A further refinement into level 3 processes exists currently (eTOM Version 4.0) only for the *Operations* process area, and no level 4 processes have been defined yet.

A Level 0 view of eTOM (see Figure 3) shows three fundamental process areas (or processes) of ICS Providers: *Strategy, Infrastructure and Product* (SIP), *Operations* and *Enterprise Management*. Four *functional areas* overlay the SIP and *Operations* process areas. Even if they do not adhere to a clearly defined layering principle, they represent the basic functions involved in service provisioning: Services are build from a combination of enterprise-own resources and sub-services sourced from suppliers and partners, and are then marketed as products to customers.

Also there are five external entities defined, with which a service provider organization needs to interact: *Customers*, *Suppliers/Partners*, *Shareholders*, *Employees* and *Other Stakeholders*. In the context of process management, these entities can be seen as external roles. The definition of roles is not extended in the higher level views of eTOM though, i.e. no enterprise-internal roles, e.g. process owners, are defined. Also – unlike ITIL – it makes no clear distinction between customers and users of a service. Even though it is mentioned in eTOM that the *Customer* entity can be refined to either *Subscriber* or *End User*, this distinction has no apparent impact on any part of the process model. eTOM’s role model can be thus described as only rudimentarily developed. Compared to ITIL, the focus of eTOM is therefore

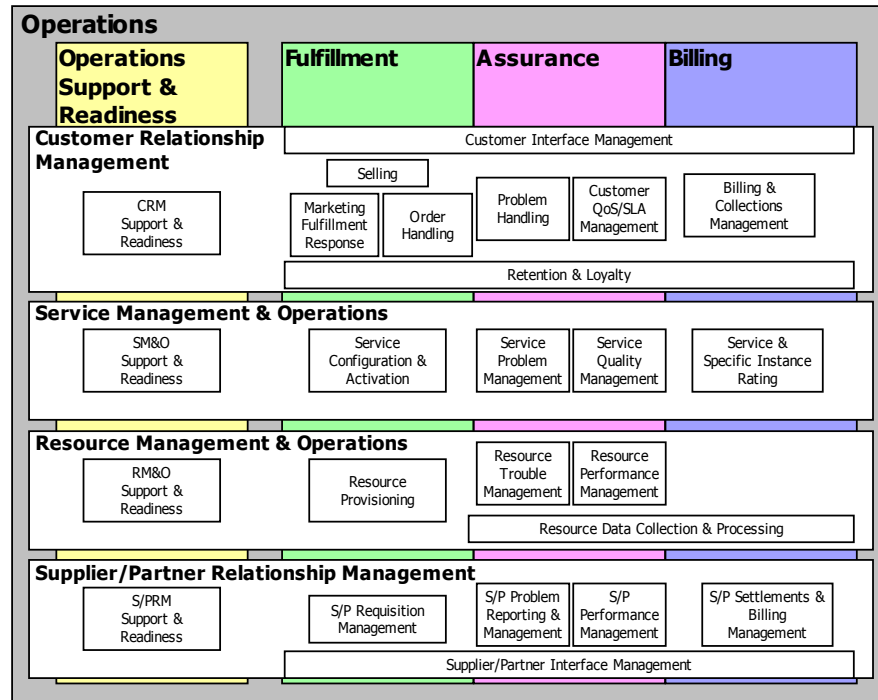


Figure 4: Level 2 Operations Processes of eTOM [2]

more on interaction between operations support systems rather than between human actors.

Operations Processes Of the process areas defined in level 0 the most relevant to service provisioning is *Operations*, which in many regards represents the core of eTOM. It covers roughly the scope of the original TOM, and contains the most refined processes - level 3 process decompositions, as well as examples of end-to-end process flows are documented only for *Operations* processes.

Figure 4 shows the level 2 view of the *Operations* processes. Each is part of one of the four level 1 processes of *Operations* (aligned to the level 0 functional areas): *Customer Relationship and Management* (CRM), *Service Management and Operations* (SM&O), *Resource Management and Operations* (SM&O) and *Supplier / Partner Relationship Management*. Perpendicular to these are four vertical *end-to-end process groupings*, which yields a process matrix - something that has become a kind of eTOM trademark.

The four end-to-end processes in *Operations* are *Operations Support and Readiness*, *Fulfillment*, *Assurance* and *Billing*. The latter three have already been present in TOM and are also often referred to as the “FAB processes” (Fulfillment, Assurance, Billing). While the functional division of *Operations* into its (horizontal) level 1 processes resembles the organization of an enterprise into departments, the division into vertical end-to-end processes corresponds more closely to the idea of business goal oriented processes. This often causes confusion. Unfortunately there is no precise, universally accepted definition of the term “business process”, but in classic business process management or process reengineering terms, eTOM’s end-to-end process groupings would be considered the real business processes, while most of what eTOM defines as processes would by many people be seen as business activities or business functions.

3.1.3 Service Provisioning in eTOM

While not being explicitly called “Service Provisioning”, it is the end-to-end process *Fulfillment* that is responsible for providing customers with a requested service in a timely and correct manner. Figure 5 shows an exemplary service provisioning workflow between level 2 process elements:

1. *Customer Interface Management* routes an order for a new service instance to *Selling*.
2. *Selling* requests and receives a priority for the order from *Retention & Loyalty* based on customer data.
3. Prioritized order is handed over to *Order Handling*.
4. *Order Handling* passes a service activation request to *Service Configuration & Activation*.
5. *Service Configuration & Activation* decides which resources are to be supplied internally and issues resource reservation requests, work orders and resource activation requests to *Resource Provisioning* - simultaneously procurement requests are issued to *S/P Requisition Management* for externally supplied resources.
6. *S/P Requisition Management* chooses external sub-providers and issues orders through *S/P Interface Management* and waits for confirmation of external resource activation.
7. *Resource Provisioning* and *S/P Requisition Management* inform *Service Configuration & Activation* when all requested resources have been activated.
8. After final tests, *Service Configuration & Activation* informs *Order Handling* about successful activation of the service.
9. *Order Handling* instructs *Customer Interface Management* to inform the customer about completion of the order.

Of course real service provisioning workflows within eTOM can be much more complex. The above example involves no presales negotiations, no service design activities, nor does it show the necessary transfer of information to initiate *Assurance* and *Billing* of the new process instance. Also, like the workflow examples provided by the TMF in the eTOM addendum F (GB921F), the above description is limited to level 2 processes.

Figure 6 shows the decomposition of *Service Configuration & Activation* into level 3 process elements. Except *Design Solution*), all these sub-processes would take part in the execution of *Service Configuration & Activation* in the above example. A level 3 workflow description of the above service provisioning example would therefore involve well above 20 process elements. Considering the planned decomposition into level 4 elements would multiply this number again.

3.1.4 Summary

The eTOM defines all generic process elements necessary for service provisioning, for each of which however it describes only the “what”, not the “how”. Thus, it does not provide ready-to-use internal or cross organizational workflows or interfaces. But the eTOM can lend a common structure to organization-specific workflows, significantly facilitate communications with customers and suppliers, and provide a basis for process automation efforts (see discussion on NGOSS and OSSJ in Section 4.5).

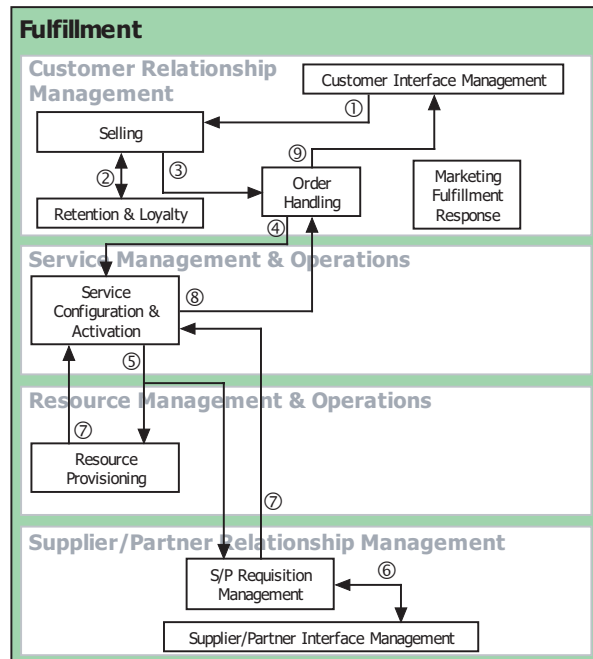


Figure 5: Simplified Fulfillment Workflow

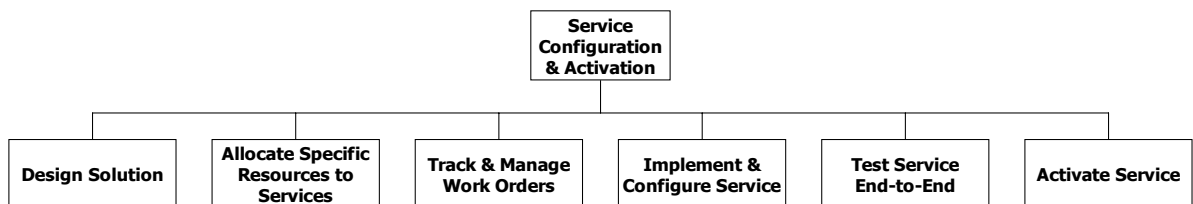


Figure 6: Level 3 Process Decomposition of Service Configuration & Activation

3.2 IT Infrastructure Library

Despite the similarity of their stated goals, the *IT Infrastructure Library* (ITIL), stemming from a data center and mainframe computer background, and the eTOM, having its roots in a telecommunications services environment, follow quite different approaches.

3.2.1 Introduction

The origins of ITIL date back to the late 1980s, when the *Central Computer and Telecommunications Agency* (CCTA) started publishing guidance for various aspects of IT operations under the label *IT Infrastructure Library*. Along with other agencies, the CCTA became part of the newly set-up *Office of Government Commerce* (OGC) in 2000, which took over ownership of ITIL. For further ITIL developments, the OGC is cooperating with the *British Standards Institution* (BSI), the *IT Service Management Forum* (itSMF), an increasingly influential association of ITIL users, and also with the two IT Service Management examination institutes, the Dutch *Exameninstitut voor Informatica* (EXIN) and the British *Information Systems Examination Board* (ISEB). The OGC is still coordinating all official efforts though, and retains

the ownership of ITIL.

While other ITIL books cover many other IT-related topics [12], the “core” of ITIL, its guidance on IT Service Management, is combined in the *Service Support* [13] and *Service Delivery* [14] volumes. Their content is also the scope for tests administered by EXIN and ISEB, to achieve IT Service Management certifications for IT professionals. Since December 2005 there is also an international standard for IT Service Management certifications of IT organizations: ISO/IEC 20000 [15, 16], an adaptation of the ITIL-aligned BS 15000 by the BSI.

3.2.2 Structure and Content

ITIL’s guidance on IT service management is documented in the books *Service Support* and *Service Delivery*, each containing the description of five processes¹ - *Service Support* additionally covers the business function *Service Desk*, which functions as a single point of contact for the *user*. Unlike eTOM, ITIL distinguishes the roles *user*, i.e. somebody using a service provided by the IT organization, and *customer*, i.e. the contractual partner of the IT organization regarding the conclusion of a *Service Level Agreement* (SLA). For the customer, the *Service Level Manager* serves as the contact person.

3.2.3 Service Provisioning in ITIL

There is no explicit “Service Provisioning” process or another process correspondent to the eTOM *Fulfillment* process grouping defined in ITIL. As a matter of fact, of all the ITIL *Service Support* and *Service Delivery* processes, only Incident Management can be more or less clearly mapped to one of eTOM *Operations*’ FAB processes, namely to *Assurance*. One of the eTOM application notes, GB921L [11] (see Section 3.1.2), gives some information on ITIL-eTOM mapping, but generally there is no exact one-to-one matching of ITIL service management processes to eTOM processes.

This is due to the fact, that ITIL and eTOM each follow implicit, yet distinct assumptions and philosophies stemming from their origins. Where eTOM, coming from a telecommunications services background, focuses on commodity or customized services offered to a multitude of customers, the ITIL, coming from a data center and mainframe background, concentrates on complex individual services offered to only a few customers (but possibly many users). Consequently, eTOM differentiates clearly between the management of services (as in service-class or product), addressed in the SIP process area and the management of service instances, addressed in the *Operations* process area - a distinction that is not made with this precision in ITIL.

As a result, ITIL processes are not concerned with the efficient, automated provisioning of multiple service instances. Where eTOM’s focus is on structuring and automating management activities for more or less standardized services, ITIL’s focus is on coordinating the collaboration of IT personnel on not completely pre-determinable tasks. ITIL processes therefore usually cannot be mapped one-to-one on eTOM processes, but have correlations to many eTOM processes, sometimes even processes from different eTOM level 0 process areas [11]. ITIL puts most weight on support and coordination of operational activities (and not the activities themselves), an area addressed in eTOM in the processes that form *Operations Support and Readiness* - which overall, even if not acknowledged by the the TMF’s “eTOM ITIL

¹Sometimes *Security Management* is counted as a sixth process of *Service Delivery*

application note” [11], have probably the highest correlation to ITIL’s service management concepts.

In summary, ITIL gives primarily guidance on processes for planning, supporting and controlling operational activities and infrastructure changes. As such, practically all ITIL-ITSM processes have impact on efficiency of service provisioning. The discussion in the following paragraphs will however be limited to presenting the core concepts and terms of the three ITIL processes of special relevance to some of the most difficult issues of service provisioning: *Service Level Management*, central to aligning the services to the customers’ needs, *Capacity Management*, ensuring a coherent approach to managing capacity issues and finally *Change Management*, helping to keep the infrastructure in a known and reliable state.

Service Level Management Service Level Management (SLM) is the process responsible within the ITIL framework for the management of interactions between the IT organization and the customer (just like the Service Desk is the single point of contact for the user). The Service Level Manager therefore acts as a middle-man, representing the IT organization in all dealings with the customer, and at the same time being the advocate of customer concerns within the IT provider.

SLM tries to achieve more customer-oriented provisioning of IT services by formulating, concluding and continuously monitoring and adapting a Service Level Agreement between the provider, i.e. the IT organization, and the customer. Figure 7 depicts the interrelationships of the most important concepts and terms central to SLM. An SLA specifies what comprises a service and how it is to be provided. In corporate scenarios (and according to ITIL) an IT service is requested by the customer organization to support one or more of its business processes. The service provider organization can itself however act as a customer, when it needs to procure sub-services for the realization of the offered service. When these sub-services are provided by an internal organization (e.g. another department within the same enterprise), an Operating Level Agreement (OLA) is concluded, an *Underpinning Contract* (UC) when the sub-services are obtained from an external provider. From the viewpoint of these sub-providers however, it is in both cases an SLA. While not part of the ITIL’s definition of these terms, OLAs usually do not include clauses about penalties to be paid for failing to meet agreed quality targets, while UCs often do.

The non-functional characteristics that a service needs to have to guarantee satisfactory support of the customer’s business process (e.g. service availability and performance) are documented in the *Service Level Requirements*. This can of course not be done by the IT organization alone, but only in collaboration with the customer. In many ways, concluding a SLA is not just subjecting the achievements of the IT to more scrutiny, it often also forces the customer the customer organization to put significant effort in defining its business goals and how these are dependent on IT services more precisely.

To fulfill these requirements, technical and organizational aspects in the IT organization need to be considered. How a service is to be realized on the technical level is documented in a *Service Specification Sheet*. This Service Spec Sheet describes what parts of the provider-owned infrastructure, as well as which UCs and OLAs are needed for the provisioning of the service. For the compliance to certain, e.g. availability-related service levels however, not only technical but also organizational aspects need to be managed. The process-oriented performance indicators that need to be met in order to fulfill service level requirements (e.g. incident resolution times, standard change cycle times etc.) are documented in a *Service*

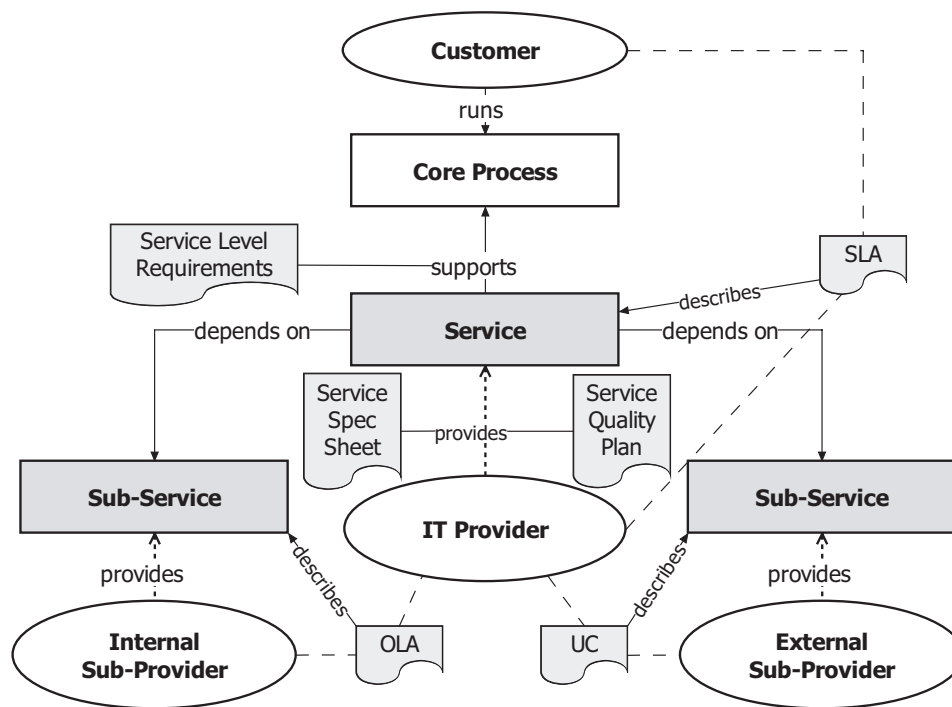


Figure 7: ITIL Service Level Management Concepts

Quality Plan.

Capacity Management *Capacity Management*, like its sibling *Availability Management*, is a planning process primarily concerned with directing infrastructure improvement efforts under a specific quality goal. The goal of capacity management is to make sure that storage, processing and network capacity are provided in a timely and cost-effective manner.

ITIL divides Capacity Management into three closely interrelated sub-processes:

Business Capacity Management: Managing the capacity requirements of the business (i.e., customer) - this involves mainly making forecasts of future capacity requirements, based e.g. on trend analysis of service utilization data or communications with the customer.

Service Capacity Management: Responsible for ensuring that the necessary tools, policies and procedures for the performance management of live, operational IT Services are in place and that collected data is recorded, analyzed and reported.

Resource Capacity Management: Ensuring effective performance management on the resource level, e.g. by establishing monitoring and reporting of CPU and network load and proactively eliminating bottlenecks before service performance is compromised.

ITIL defines nine principal activities of Capacity Management: *Monitoring, Analysis, Tuning, Implementation, Storage of Capacity Management Data* (into a *Capacity Management Database*), *Demand Management, Modeling, Application Sizing, Production of the Capacity Plan*. Even though Capacity Management is called a process by ITIL, these activities are not arranged in a structured workflow.

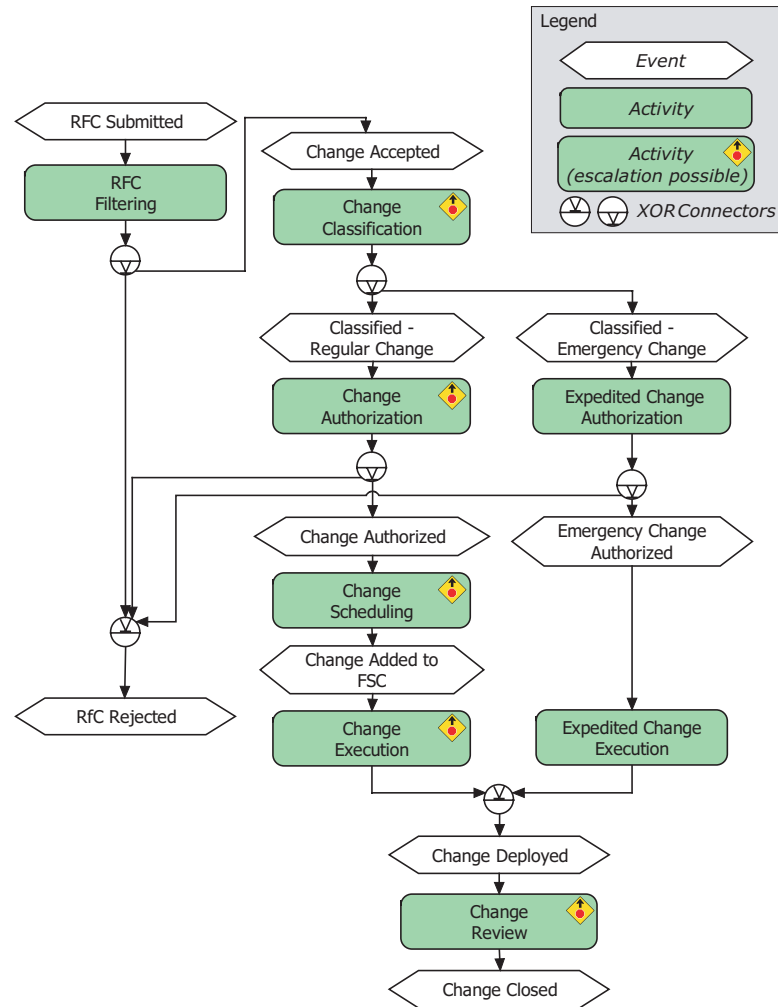


Figure 8: ITIL Change Management Workflow

The activity in ITIL’s Capacity Management that is probably the most neglected by resource-oriented performance management approaches is Demand Management. This refers to supporting the match of capacity requirements and provided capacity, not solely by increasing the latter, but by trying to influence capacity requirements through influencing user behavior. For example, if peak demand can be influenced, e.g. by a lower price for service usage during off-office hours or establishing agreements that non-critical batch jobs should not be started on Fridays when system load is the highest, then peak demand and thus capacity requirements can be reduced.

Change Management ITIL Change Management tries to ensure that changes to the IT infrastructure are executed with the highest possible efficiency and lowest possible impact on service quality. For this purpose, all changes falling into the scope of change management are put under the control of the change management. What falls into change management’s scope, i.e. what constitutes a “change” in the IT organization, needs to be defined. This is usually done aligned to the detail level in which the infrastructure is documented in the *Configuration*

Management Database (CMDB) or consequently the Service Specification Sheets. The level of detail required here is usually not very high, as “change” in an ITIL context relates to structural infrastructure changes (upgrades of hardware, software updates and the like), rather than an altered operational status (e.g. a change in the value of Management Information Base/MIB counters).

Of the three ITIL processes discussed here, Change Management is the only one for which a concrete workflow can be defined [17]. The basic Change Management workflow is illustrated in Figure 8 by an Event-driven Process Chain (EPC) diagram, a graphical notation used in the ARIS framework [18] for modelling workflows. For every change (except for *standard changes*, see below), a *Request For Change* (RFC) must be submitted to change management staff, i.e. in smaller scenarios the change manager. RFCs are screened if they comply to formal guidelines that are laid down by change management (e.g., are name, department and position of the requestor given?). After that, changes are classified according to possible impact and their urgency. At this point, there is a distinction between very regular change procedures and expedited emergency change procedures for very urgent, critical changes (e.g. taking the whole system offline to respond to a major intrusion).

The following describes regular change procedures first. All changes need to be authorized – by whom depends on the classification of the change, e.g. change manager or change management staff for minor impact changes, *Change Advisory Board* (CAB) for changes with significant impact, up to the inclusion of CAB and business management (e.g. CEO) into an executive committee for the authorization for major impact changes. Once they are authorized, they need to be scheduled and executed under the control of change management. Finally every change should be subject to a *Post Implementation Review* (PIR), evaluating if the goals of the change (e.g. availability improvement of a specific service) were met. Depending on the nature of the change and how achievement of the change goals can be measured, the PIR can happen days, weeks or even months after the change has been executed.

Any of these activities might require a reclassification and consequently a reauthorization of the change. For example, a patch to a server software package might have been classified as a minor change and authorized by the change manager. If then it becomes apparent during testing (a sub-activity of change execution), that not all dependencies had been considered and that for instance the patch necessitates an operating system upgrade or has other unforeseen side-effects, then the change needs to be reconsidered. If, for example, the needed operating system upgrade is considered a significant change, then the CAB must be convened, i.e. the change needs to be escalated.

Such an escalation is not possible for emergency changes. Time is of the utmost importance in emergencies, but emergency change procedures still go through the same process, albeit in an expedited way. An emergency change still needs to be authorized. In practice, emergency changes by their very nature are high risk, thereby of the highest impact classification and warrant the summoning of a CAB meeting. However since there usually is not enough time to hold a regular CAB meeting, the authorization of emergency changes is transferred to the CAB-EC, the *Change Advisory Board Emergency Committee*, constituted of all potential CAB members that can be summoned on short notice. Since the CAB-EC includes all authority that can be assembled within the time constraints, there cannot be any escalation – there simply is no higher authority available at that moment. Also, since action for an emergency change must be immediate, there is no need for change scheduling. Otherwise, the process for emergency changes still includes all change management activities (in an expedited way), as well as a regular Post Implementation Review.

Specific types of changes that are frequent, whose impact is well known and low, and for which detailed procedures can be established, can be declared pre-authorized *standard changes*. Typical examples might be the replacement of a hard disk in a RAID-system, replacement of a network interface card or other standard replacement activities. Such standard changes still go formally through the change management process, but usually without much involvement of change management staff. There is no authorization activity, and scheduling, execution and review can often be performed by system administrators according to documented procedures without explicit control by change management staff.

If the provisioning of a service constitutes a change it will in practice be dependent on the type of service and how it is realized: commodity and customized services should be implemented in a way that creation of a new service instance requires only standard changes, at least as long as capacity permits. Individual services however, will often require more complex changes. An institutionalized change management process based on ITIL guidance can improve the coordination of all stakeholders in such a change and make the results much more predictable.

3.2.4 Summary

While on the surface, ITIL and eTOM share a common goal, their approaches to delivering a framework for IT service management processes are quite different. While eTOM aims for improving efficiency by automation and tries covering the full scope of all provider processes, ITIL sacrifices comprehensiveness for more detailed guidance on “pain areas” and sees quality improvement as its main goal. For a general discussion of eTOM and ITIL see the chapter by Scheffel and Strassner in this volume [19].

3.3 Service Oriented Architectures

Service Oriented Architecture (SOA) is a concept for designing IT architectures, also often cited as a new approach for addressing *Enterprise Application Integration* (EAI) issues. While it is often referred to in the context of Web Services, it is technology-independent [20]. There are still many different notions about what constitutes a SOA. As yet, there is no standard or even commonly accepted definition of SOA. A very generic definition is provided by the *Organization for the Advancement of Structured Information Standards* (OASIS), an international consortium for e-business standards, describing SOA as “a paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains” [21] - this however describes rather the goal of SOA than the concepts with which to achieve these goals. The OASIS is however also working on a reference model, which will hopefully create a framework to enable better description and specification of SOAs.

Despite the lack of a universally accepted definition, there are some basic ideas and concepts that are commonly attributed to SOA:

Making functionality network-accessible The idea is to make more of the functionality provided by an IT system, previously often dedicated for a specific application, accessible over the network through syntactically standardized interfaces. Compared to objects or software components, services are thought to be usually more “coarse-grained”, i.e. a service should be able to address use cases outside the system in which it is realized. A description of the service and information on how to access and use it is then published in a service repository. This

facilitates the reuse of software and system components for purposes that were not anticipated at design time.

Service composition and loose coupling Published, accessible services can be used for building new services through composition. The thinking is that in general, services should be loosely coupled and communicate in an asynchronous manner. This means for instance, that service invocations should be executed in a message-based, rather than a remote procedure call (RPC) style manner, i.e. the requestor will not be blocked until receiving a service response and no communication context needs to be maintained. Also, with no central components and services – in principle not needing to share anything but a network connection and a possibly very simple message exchange mechanism like SOAP over HTTP – cross-organizational service interaction can be realized rather easily.

Compared with previous paradigms for distributed architectures like distributed object models, SOA strives to predetermine only a minimal set of rules for the communication between services (or components). Services could still be realized using object technology, but interaction between services is not dependent on a shared object model, programming paradigm or even type system. SOA is no silver bullet for all problems, as this new flexibility often comes at the cost of lost functionality provided by tight coupling middleware, e.g. reliable messaging or security mechanisms.

The SOA idea holds great promises for the future of provisioning application layer services: Integrating services across different organizational domains, easier load-balancing and failover mechanisms (possibly across organizational boundaries), even automated service discovery and composition - all these are SOA visions.

From a service provisioning perspective, a SOA infrastructure could offer standardized interfaces for service invocation and configuration, thus providing in effect a management interface to the building blocks of more complex, compound services. Though usually request/response software services are described as service examples, services in the SOA sense can also make “real world”- functionality accessible. So activation and configuration of a network service, e.g. a VPN, could very well be published as a service and integrated into a SOA. In effect the NGOSS architectural guidelines (see also Section 4.3), first published well before the term “SOA” became popular, aim for such a modular, interface-oriented management system.

There still is a long way to go, but given the substantial industry interest in SOA, it is clear that it addresses important needs and as a concept will probably stay the focus of many efforts in distributed systems - maybe even when the current Web Services protocols have long been superseded.

4 Trends in Technologies and Tools

A variety of tools are already available to support service provisioning processes. In this section state-of-the-art tools are grouped according the service provisioning phases and analyzed with respect to their contribution to the challenges outlined in section 2. Beside of tools for specific phases, there are workflow tools supporting the whole service provisioning process which are presented in subsection 4.5. Due to the amount of tools being offered only a set of selected tools is presented in the following.

4.1 Service Planning

The service planning phase is quite diverse depending on the background of the service to be provided. Usually, a service is described in an abstract manner focusing on its desired functionalities. In addition to performing a market survey to get an estimation about the sales price and the potential number of customers, the provider also has to get an estimation about the costs for the service provisioning using information from sources like inventory management tools, staff salaries, etc. Providers have often developed home-grown solutions for this by retrieving and processing data from SAP-like systems. However, it can be witnessed that the diversity of service planning which is closely tight to the positioning of the provider in the business market makes it difficult to offer standard support tools.

4.2 Service Ordering

In the service ordering phase the details of an order are negotiated with customers. According to the kind of scenario (individual services, customized services, commodity services), this negotiation can be complex or simple. Nevertheless, the service provider has to make sure throughout all provisioning steps that the service order can be fulfilled. Therefore, it has to be verified whether there are currently enough resources or can be made available until the desired fulfillment date keeping in mind the QoS guarantees. In addition, a successful service order has to be registered and transferred to the service implementation. For example, the new service contract has to be added to the accounting process.

Tools which can be applied for this phase comprise web shops (being constructed with tools like Intershop) or home-grown solutions such as web front ends communicating with SAP installations. SAP is frequently used to store customer data and their SLAs.

4.3 Service Fulfillment

During the service implementation, services and resources have to be ordered from suppliers and configured according to the provisioning requirements. These steps should be automated in order to efficiently deliver the services. Tools should be applied to decide about the sharing of resources and to automatically generate resource configurations (hardware and software). For doing so, it is useful to have a solution database storing resource configurations which have been applied for previous service requests.

4.3.1 Resource Virtualization

During the service implementation the decision about the sharing of resources is very much dependent on the kind of resources in question. For the network it is common practice not to use dedicated hardware, but to create virtual networks using technologies like Multi Protocol Label Switching (MPLS) and/or Virtual Private Networks (VPNs). Hardware components like servers can also be virtualized (e.g. in Web hosting applications) such that these components can be used by customers as if they had dedicated hardware. Another common practice is to share storage space among customers again giving them the impression they are using dedicated resources. Obviously, the implementation of the virtualization has to protect the integrity of each customer. In addition, it is possible to run another operation system on top of the basic operation system which can be helpful depending on the desired software applications. In some cases the virtualization possibilities are already included by vendors. SAP installations can,

for example, be shared between different parties and Checkpoint also offers virtual firewalls where a real firewall can filter packets for different target networks.

Customer requirements especially in the area of individual service provisioning have a large influence on virtualization decisions. A virtual firewall, non-dedicated DNS servers, and shared mail servers may be acceptable for customers with small budgets and less strict security requirements, while other requirements may lead to dedicated and consequently much more expensive solutions. In contrast to network sharing and use of tool intrinsic mechanisms, it is currently not usual for service providers to add sharing mechanisms to resources on their own.

4.3.2 Tool Support for Resource Configuration

Starting from its traditional network management software HP [22] is currently offering a variety of tools to support the service provisioning with respect to ITIL. These tools comprise solutions for service desks and inventory management including an automated configuration detection which is especially useful for Windows Environments. *IBM Tivoli's* management solutions have gone through a similar evolution as those from HP. IBM also fosters the support of ITIL processes and offers a dedicated tool for service provisioning which is called *IBM Tivoli Provisioning Manager* [23]. It automates the provisioning and configuration of servers, virtual servers, operating systems, middleware, applications, storage, and network devices. This is done by using preconfigured automation packages which are available for a number of products (e.g., *Citrix*, *MS SQL server*, *Siebel*). An extension to the *Provisioning Manager* for further automation is the *Intelligent Orchestrator* supporting a scheduling of the automation package installation. Similar to HP and IBM, CA also offers a comparable set of tools.

For software configuration management, Microsoft offers suites for the automated deployment of software in enterprise management scenarios (*Systems Management Server*) and for application management (*Operations Manager*). *Sun N1 software* [24] is designed to support the service lifecycle. Its component *Service Provisioning System* allows for application service provisioning by remotely installing operation systems (Solaris, Linux, Windows), use of XML descriptions to deploy, upgrade, delete, start and stop applications, and record administrator action (allowing for a possible rollback). *Cisco's Configuration Engine* [25] is useful to deploy equipment at customer locations. Its template-based definition of deployment procedures makes it possible to preconfigure equipment so that these configuration steps do not have to be performed by provider staff. In sum, it can be witnessed that tools are already available to ease the configuration of hardware and software. However, the tools are often tied to equipment vendors so that their application may pose additional constraints for the service implementation.

Cfengine [26] is a site configuration engine, that is designed to let administrators use configuration files to describe the desired state of a system, rather than describing what should be done to a system to reach that state. The idea of *Cfengine* is to create a single set of configuration files which describe the setup of all hosts on a network. In a typical setup, *Cfengine* runs on every host, retrieves configuration data from the master server, and applies changes as needed.

4.3.3 Web Service and Grid Service Standards

A service can be implemented using the Service Oriented Architecture paradigm. This can be useful if the service can be split up into separate tasks being realized by different groups

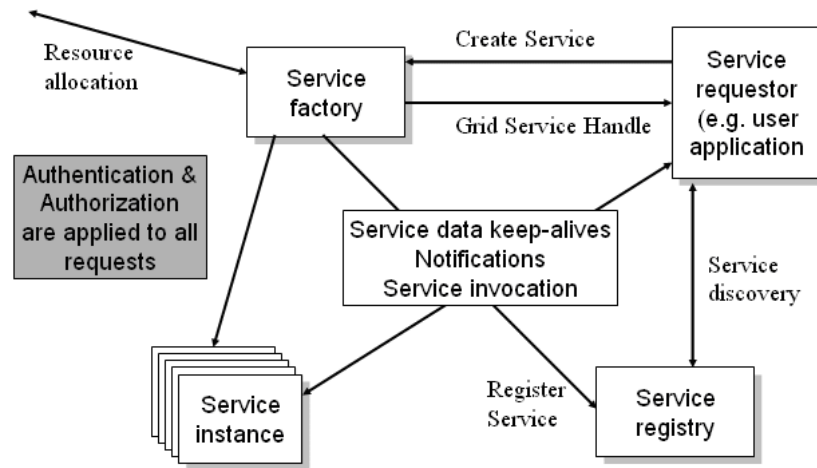


Figure 9: Grid Service provisioning [28]

(either internal or partially by external providers). A popular means for the realization of SOAs are *Web services*. These services are specified in an evolving set of standards by *W3C Consortium*, *DMTF*, *Open Grid Forum*, *OASIS* and other standards bodies. Web service specifications deal with the platform and programming language independent specification of autonomous services which can be used to collaborate as distributed applications. They can be described, published, discovered, orchestrated and are designed for the implementation of business processes especially across organizational boundaries.

The most popular Web services standard is *WSDL (Web Services Description Language)* for the specification of Web service interfaces. Other standards deal with the specification of service registry, security, policy, service orchestration, and service agreements. Furthermore, there is an evolving standard for the management of Web services.

For service provisioning in particular an XML-based language called *Service Provisioning Markup Language* [27] is under development aiming at the support for the whole service life-cycle. However, the language is tied to the specification of resources as Web services.

Current problems in using Web services are missing semantics for the service attributes, new challenges for fault management among independent services and a missing concept to deal with the service lifetime.

In the area of Grid computing the *Open Grid Service Architecture (OGSA)* [28] aims at automating the provisioning of so called *Grid services*. Grid services can be regarded as an extension of Web services as they are compliant with the WSDL standard, but also define some more features in a standardized XML schema. These features include service states for automated service instantiation and removal.

Figure 9 shows interactions which are defined for the Grid service provisioning (*Web Services Resource Framework (WS-RF)*). A Service requester would like to create a Grid service by contacting the Service factory. The Service factory composes resources to provide the service and returns an access handle to the Service requester. The requester can then communicate with the Service instance for the exchange of service data, keep-alives, notifications and for invoking the service. In addition to the service creation on demand, the requester may also contact the service registry where services have been registered by the Service factory for reuse

by other requesters.

An important extension in WS-RF in comparison to Web services is the explicit creation of services which is part of the infrastructure while this happens in an unspecified manner for standard Web services. This extension has been made to allow for the creation of transient services which may for example be created and only exist to handle a long-term transaction.

The *Configuration Description, Deployment, and Lifecycle Management Specification* (CDDL) by the Open Grid Forum (former Global Grid Forum) addresses how to describe the configuration of services, deploy them on the Grid, and manage their deployment lifecycle. The document GFD.50 serves as an informational foundation document, while GFD.51 [29], GFD.65 [30] and GFD.69 [31] contain the recommended SmartFrog-based language specification, component model and deployment API, respectively.

In summary, SOAs and Web/Grid services are beginning to play a role in the realization of services in the industry. While the concept of SOAs can be presumed to remain for the longer term, the standards for Web/Grid services are still changing/evolving and may undergo larger revisions. In [32] one of the first realizations of a Web services-based service provisioning in a large scale scenario in the industry is shown.

4.4 Service Operations

In the service operations phase tools are needed to manage the configuration of services and resources, manage the customer SLA data, collect data for SLA reporting and accounting purposes, and to interact with the customers. The latter allow the customers to change the service configuration in a predefined manner as well as to retrieve accounting and SLA reports.

Since the discussion of service operations is out of scope of this article, we are not going into tool details here. However, it can be concluded that a variety of tools exist to support these processes. For the configuration management those tools which have been mentioned for the service implementation support also offer solutions here. The configuration itself can be stored different kinds of databases, which often claim to be compliant to ITIL's CMDB, or (more seldom) in LDAP. Vendors like Mercury Interactive offer tools for SLA reporting.

4.5 Service Provisioning Workflow Support

For the whole process of service provisioning a variety of steps have to be carried out. For following these steps, process management tools can be applied including the definition of conditions which have to be fulfilled for completing a step.

As ITIL has become a very popular standard for service management, many vendors strive to provide ITIL compliant tools as mentioned for HP and IBM. As the ITIL guidelines are written in prose, it is however difficult to define what this compliance actually means. In addition, these tools only address a subset of the ITIL processes.

The Operation Support System through Java (OSSJ) [33] initiative aims at the implementation and deployment of reusable OSS systems using Java technologies. It has provided its own open API specifications, reference implementations and test kits, while commercial products implementing the specifications also became available. In the reference implementation OSSJ uses J2EE to provide strongly-typed, loosely-coupled APIs using XML over Java Messaging Service. Just as important, it applies the Java Community Process for standardization of APIs. In 2004 OSSJ and TeleManagement Forum's *NGOSS* [7] started a collaboration since both initiatives can be regarded as complementary. NGOSS provides technology-neutral

business-process and system frameworks, UML information models and resources for building NGOSS-compliant OSS systems and components, while OSSJ provides an OSS implementation in Java.

SAP tools are used by many companies for the support of a variety of business processes. Other tools which are applied in the service provisioning process access SAP to retrieve information needed for the service provisioning. SAP also supports business processes directly by allowing for the modeling of processes using the ARIS business process notation [18]. Other interesting tools in this context comprise Microsoft BizTalk and Oracle's BPEL Engine.

4.6 Summary

The typical situation in service provisioning can be summarized as follows. A provider usually has a set of home-grown and commercial tools which are applied to service provisioning. These tools are primarily applied to service implementation and operation and each one of them is focused on a specific task. Therefore, the collaboration of these tools is often cumbersome in practice leading to problems like data duplication and inconsistencies. Often, workflows are defined for the service provisioning, but their execution is dependent on the staff and the general service provisioning circumstances.

5 Research Activities

As the presentation of concepts and frameworks in Section 3 as well as the examination of available tools in the previous section showed, a lot of issues in the context of service provisioning cannot be regarded as satisfactorily solved. Therefore, research being conducted in academia and industry is presented in this section which addresses some of these challenges. The presentation is ordered with respect to the service provisioning phases.

5.1 Service Planning

In some areas of service provisioning such as mobile phone services or context-aware services, a frequent change of the service offers can be witnessed. Therefore, research efforts are aimed at allowing the rapid definition and planning of services, especially to react to changes in the market.

Service profiles Clemm et al. [34] proposed the definition of services profiles which denote a technology-neutral description of services using a template. It includes a set of features that a service may have and for each feature a set of possible parameters. For the later service instantiation, technology-specific provisioning profiles are used to allow for the implementation of the services using different technologies. A service provisioning engine was prototypically implemented and applied for two example scenarios (residential subscriber services, Internet data centers).

5.2 Service Ordering

While the service ordering is quite simple for commodity and customized services, it may require a series of interactions between customer and provider for individual services. The

implementation of a CSM as described in the following can be an important step to support the required interactions.

Customer Service Management Langer and Nerb [35] provided a detailed concept for implementing a *Customer Service Management* (CSM) interface for a given service provisioning scenario. CSM aims at allowing the customer to access provider-internal service management information and to self-manage subscribed services in a predefined manner. With respect to service provisioning the CSM interface allows the customer to order new services or to change parameters of currently subscribed services. The implementation of CSM by e.g. providing a web-based interface allows for the processing of customer change requests since these requests are given in a standardized format.

A central step in service ordering is the negotiation of SLAs. This negotiation requires to know which kinds of SLA mechanisms and guarantees are suitable to allow for stable service operations. Begnum et al. [36] showed that policies using non-linear feedback mechanisms can result in quite complex behavior which may cause the repeated downgrading of service levels when not parameterized with care.

The negotiation phase for service provisioning is quite different in peer-to-peer environments. Here, the negotiation of trust is often a crucial issue since the peers do not have a priori knowledge of each other especially in large-scale deployments. For enabling the service provisioning despite of limited trust, Burgess [37] proposes to accept different trust levels depending on the service's requirements.

In [38], Sarangan and Chen give an overview of protocols for dynamic service negotiation of network services.

5.3 Service Fulfillment and Operations

Even though some tools are already in place for service implementation, there are still open issues which need to be addressed in a general manner.

Service templates and CBR In her postdoctoral thesis, Dreo[39] proposes the application of Case-Based Reasoning (CBR) techniques for service provisioning and especially addresses the question how to obtain the requested resp. desired granularity of fulfillment, operational and withdrawal steps. The idea is to describe a service as a *service template*, including service ordering information, the necessary fulfillment and operational steps, the withdrawal steps if the service needs to be removed from the system as well as the policies associated with this service. Having such a service template structure which corresponds to a *case structure*, enables us to apply the CBR paradigm to dynamic service provisioning. The main idea is to think of a *service template as a case*. If a new service is ordered by a customer, then the service template library (case repository) is searched for a similar service order, resp. service template. After a similar service template is found, it is retrieved and adapted to the specific customer requests (specified in the service ordering), and afterwards the adapted steps are executed. The proposed application of the CBR paradigm successfully addresses the following problems: (i) to gain the requested granularity of steps (as precisely as possible) and (ii) to ensure up-to-date provisioning steps since these need to be updated due to changes in the infrastructure, customer requests etc.

CHAMPS Keller and Badonnel [40] describe IBM’s research system *CHAMPS* (*CHAnge Management with Planning and Scheduling*) which is prototypically applied for application service provisioning. It consists of two major components: *Task Graph Builder* and *Planner & Scheduler*. The Task Graph Builder is designed to build a workflow for performing a change in service operations. This workflow is described using *BPEL* (*Business Process Execution Language*) which is a popular standard for the description of processes in the context of Web services. The Planner & Scheduler component receives the workflows from the Task Graph Builder and generates a schedule for executing the changes under consideration of a set of constraints. These constraints prevent conflicts in the change execution due to resource access conflicts and time conditions as well as non-technical requirements such as SLAs and policies.

Quantifying complexity Brown et al. [41] provide a model to quantify configuration complexity, identifying execution complexity, parameter complexity and memory complexity as the major components of the model. The model is then applied to the CHAMPS change management system mentioned before to show the reduction in human-perceived configuration complexity and installation time.

Diao and Keller [42] address complexity quantification in IT service management. Their goal is to measure the savings in process complexity, when processes are re-engineered or automated. The authors propose a process model based on roles and their participation in tasks producing or consuming business items. They further propose quantifying process complexity along the dimensions execution complexity, coordination complexity, and business item complexity. The authors also describe a tooling concept and a brief evaluation.

Quartermaster HP’s *Quartermaster* research project [43] aimed at automating data center management tasks, primarily focused on service fulfillment. The core of the system is a CIM model repository representing four layers: service, application, system, and physical and allowed the definition of policies. A tool chain is arranged around the core providing management functions. Tools include functions of resource allocation and assignment, deployment and run-time control. Automation is achieved by orchestration of tools through workflows. A fundamental concept of Quartermaster is a resource topology. It is a formalized representation (model) of a configured resource environment supporting an application. An editor is offered to design resource topologies. They are stored in the core repository and passed through the tool chain for instantiation. During operation, so-called Service Delivery Controllers, as part of the same tool chain, continuously compare the currently provided service quality with a desired state and re-adjust service operations in a closed control loop.

Model translation Sahai et al. [44] describe the automated translation of resource models, generated by a constraint solver in Quartermaster as MOF models based on CIM, into scripts for software deployment and configuration based on SmartFrog. They address “Built-to-Order” systems, which would well fit in the category of customized services and most individual services. In a follow-up paper [45], automated staging and testing are included in the provisioning workflow. Solution designers can get data about the performance of the deployed solutions via instrumented applications and monitoring tools. In an open-loop like fashion, they can now tweak the input to the constraint solver as well as policies and re-iterate the process. In a deployment of a TPC-W scenario, the performance of several solutions based on servers with different capabilities is compared.

SunTone *Sun's SunTone* program [46] provides an open service model for service delivery especially targeting to avoid “data silos” (isolated data repositories which lead to data duplication). The model is a 3-dimensional cube with the dimensions *tiers - application partitioning*, *layers - hardware/software stack*, and *systemic qualities*. The initiative offers the certification of IT or business processes concerning compliance with the model.

In the service implementation resources have to be allocated to provide the services. As this should be done in an efficient manner, several theoretical approaches have been devised. A promising approach is to model the resource allocation problem as a constraint satisfaction problem [47] for which a set of efficient algorithms exist. This approach can deal with incomplete data and can cope with changes in highly dynamic environments.

For the interaction of resources Burgess [48] proposes to use the *promise theory*. This theory from the area of autonomic computing is based on modeling the resources as autonomous entities. The service fulfillment and operation is therefore a result of the collaboration of independent entities. This theory is in particular interesting for virtualized resources like virtual servers.

5.4 Information modeling

Since the typical service provisioning that is found within service providers usually implies the distribution of service related information into various data repositories, it is often difficult to get an overview of the service provisioning effectiveness and efficiency. In addition, the data duplication can easily lead to data inconsistencies.

Service MIB Sailer [49] proposes to extend the construction of MIBs, which are usually defined for network and systems management, with respect to services. A *Service MIB* should comprise all information that is needed for the service provisioning including information from all FCAPS (fault, configuration, accounting, performance, security) functional areas. The information modeling for the *Service MIB* is carried out using object-oriented design patterns and by defining a language for the specification of service instances.

For the introduction of services a model-driven approach is proposed in [50] where the composition of components for the service creation is done via a construction GUI. The developed research prototype called *Service Designer* allows for the design of generic service models and the creation of OSS-dependent views. It has been prototypically applied to VoIP and push-to-talk services.

Eilam et al. [51] describe a model-based approach to manage the configuration complexity of distributed applications in data centers. It builds on the principle of separation of concerns. Humans with different roles, such as developers, deployment experts, domain experts, data center operators and business users provide their knowledge as models. These models are taken into account by management components like deployment topology generator, optimizer, planner and provisioning engine. The method is applied to the deployment and configuration of J2EE-based web applications.

6 Service Provisioning in Practice and Operations

After service provisioning had been analyzed from the viewpoint of challenges and process alignment in the previous sections, this section is about how service provisioning is currently realized in real provider environments. After giving an overview on current best practices and how they address the challenges listed in Section 2, outlines of interviews with practitioners give some insight into their view on service provisioning. The focus of the questions was on process organization and tool support for service provisioning. We conclude this section with two specific scenarios.

6.1 Best Practices in Resource Allocation and Assignment

Certain patterns can be witnessed in the allocation and assignment of resources with many service providers.

6.1.1 Late assignment of resources to requests

Service providers can distinguish between resource *allocation*, keeping track of the number of used and spare resources, and resource *assignment*, keeping track of the specific resource instances assigned to the requests. Resources can be allocated, when a new request is received, which needs resources at a future point in time. Specific instances are assigned shortly before the request comes due. This way, the service provider saves assignment efforts, that would have to be redone, when resources fail or requests change. In addition, resources can be allocated, that have not even been delivered by the supplier yet. This best practice works in favor of reducing management efforts and adapting to changes.

Example: Imagine that a service request requires one server after three months time for a period of two weeks. Allocation deals with capacities, and therefore would reduce the available capacity in the requested time frame by one server in an allocation calendar. A particular server would be assigned to the request shortly before service activation instead of when receiving the request.

6.1.2 Soft-state assignment of resources to requests

To adapt to changes and to reclaim resources, resource assignments should regularly be reevaluated. This can be enforced by the service provider by associating a lease time with the assignment of resources to requests. When the lease time is over, the customer is asked to refresh the request.

Soft-state assignment is typically used when providing services that the service provider is not billing for. Due to the lack of obligations, customers can forget about the resources assigned to them, without explicitly expressing that they no longer use the services. Over time this leads to a high percentage of stale entries in data stores that save information on resource assignments, no longer reflecting actual service usage.

Example: Service providers use soft-state assignments in their daily operations for many types of user accounts, like for mail, databases, and remote login, and IP addresses. Soft-state assignment is also a typical component of protocols that control resource assignment such as Dynamic Host Configuration Protocol (DHCP) and Resource Reservation Protocol (RSVP).

6.1.3 Configurable resources with remote management support

All but the simplest resources need configuration to adjust the functionality of the resource to the service provider's individual requirements. The amount of configuration options therefore reflects the flexibility of the resource. In conjunction with resources typically being distributed over several locations, this necessitates devices that can be managed remotely providing an appropriate management interface. The use of management systems, so far mainly for monitoring and remote configuration of devices, requires these management interfaces to comply with well-established standards to ensure interoperability of management systems and managed components. Standards-compliant management interfaces also provide the basis for automating the provisioning process.

Configurable resources with remote management work in favor of fast provisioning, adaptation to changes and scalable provisioning. At the same time they increase the complexity of resource descriptions due to their large number of configuration parameters and options.

Example: Managable network components like routers and switches include standard remote management interfaces accessible via Simple Network Management Protocol (SNMP). In addition, they provide command line interfaces (CLI), accessible locally via a serial console port or remotely via telnet or secure shell. While the CLIs are vendor-dependent, vendors in turn work hard to keep a certain common set of command syntax and semantics throughout their product lines.

6.1.4 Resource virtualization

Resource virtualization techniques have been developed to ease changes in the mapping of logical resources to physical resources without too much increase in visible management complexity. They provide the view of dedicated resources in a shared-resource environment to the users of the virtual resources, aiming at standardization and reducing total resource consumption. A considerable amount of complexity in terms of code and models is put into the virtualization frameworks and therefore hidden from the user and administrator, while some configuration options are exposed via a management interface. See [52] for a more in-depth investigation of the impact of virtualization on management systems.

Resource virtualization can have a mixed impact on provisioning and operations. On the one hand, it can reduce operational complexity by enabling functions that are hard or impossible to achieve on physical resources, e.g., snapshots of virtual servers that can easily be installed on remote servers or adjusting the size of logical volumes at runtime. On the other hand, when things go wrong, virtualization adds a layer of potential reasons for errors, thereby complicating troubleshooting. Finally, the level of influence and security provided by logical separation instead of physical separation in virtualized environments can be hard to assess, leading to mutual influence in shared resource pools.

Example: Resource virtualization is used in many ways, such as:

- Virtual server software, like VMWare, Xen, and Virtuozzo, create multiple logically separate virtual servers on a single physical server.
- Virtual hosts in an HTTP server allow for virtual web servers.
- Java's Virtual Machine (JVM) decouples applications from the specific processor and operating system architecture.

- Logical volumes in Storage Area Networks (SAN) can be mapped freely to spare hard disks.
- Multi-customer capable systems run service instances for multiple unrelated customers on the same set of physical resources with logical separation.
- Virtual local area networks (VLAN) decouple broadcast domains from the physical network topology, enabling topology changes by remote configuration instead of manually patching cables to different interfaces.
- Techniques, like virtual paths and channels in Asynchronous Transfer Mode (ATM) or label-switched paths in conjunction with the Resource Reservation Protocol (RSVP) in networks utilizing Multi-Protocol Label Switching (MPLS), partition end-to-end link bandwidth to create virtual dedicated network links.
- Virtual private networks (VPN) provide logically separated networks in a shared network topology.

6.1.5 Overprovisioning

When purchasing resources or subscribing to subservices, a precise requirements specification and in-depth performance evaluation is often hard and costly for the service provider mainly due to uncertainties in future resource utilization. Therefore, service providers make educated guesses and add a fair amount of extra headroom to deal with uncertainties as well as to remain flexible to address unpredicted demands or to have room for expansion.

Overprovisioning is therefore a way to keep enough resources for changing or unpredicted demands. It provides flexibility and room for expansion, so that new service requests can be deployed quickly to spare resources.

In addition, overprovisioning reduces the frequency of needed changes. As changes are one of the main reasons of service failures or disruptions, overprovisioning reduces potential service downtimes. Monitoring data such as workload profiles should be regularly analyzed to have a realistic view on resource utilization and the true amount of headroom.

A special form of overprovisioning is assigning dedicated resources to certain services or customers to work around QoS management in shared resource pools even though predicting in advance that the resources will be underutilized. This strategy can make sense for low-cost resources as the win in a reduction of operational complexity, e.g., no need for Quality of Service management, direct mapping of services/customers to particular resources and the other way around in incident and problem management can outweigh the loss resulting from having to provide more resources. With new technology and management systems on the horizon, which may provide resource sharing with only marginal additional management complexity (see “Resource Virtualization”) or better scheduling capabilities, this strategy should be regularly verified by market research, though.

Overprovisioning loosens requirements on choosing the most appropriate set of resources and therefore greatly simplifies the provisioning process at the cost of increased overall resource consumption.

Example: Overprovisioning of resources is typically cheap when introducing new hardware resources as the cost of labor and operational disruption is about the same regardless of the type of new hardware resources and the additional price for more

capable hardware resources (e.g., fiber optics with a higher nominal bandwidth, more powerful servers, etc.) is typically minor compared to other costs.

6.1.6 Oversubscription

In many scenarios, customers do not make use of the full share of resources that a peak usage of the subscribed service would require. If planning resource consumption for peak service demand for every customer, resources would be underutilized. In contrast, planning the provisioning for average customer demand allows to provide services to additional customers and therefore increases profit due to increased resource utilization.

Oversubscription works in favor of increasing resource utilization, but leads to challenges of QoS management in shared resource pools. Statistical models as well as a continuous monitoring of the actual resource utilization level are needed in order to avoid SLA breaches.

Example: Statistically, not all 500 DSL subscribers connected to a single DSL access multiplexer (DSLAM) download data using the full downstream bandwidth, e.g., 2 Mbit/s, at the same time. While the downstream link from the next router to the DSLAM may only have a bandwidth of 155 Mbit/s (in other words 77x max individual downstream bandwidth), typically all actual concurrent service users may experience a good service quality at full download speed.

6.2 Comments by Practitioners in Service Provisioning

This section outlines the main findings of three interviews with solution engineers from a large provider of enterprise services. They were asked to comment on the current state of the art in service provisioning in actual daily operations, mainly with respect to processes and tools. The gist of the interviews was compiled into short paragraphs. It is interesting, that the three engineers did not agree in their statements although they work for the same company.

Comments of the first engineer. Processes for service provisioning are defined in the industry at such a level of detail that a hardcopy would fill walls. Therefore, people in daily solutions business do not pay too much attention to these processes since they are often hard to understand and follow from the perspective of each employee. An exception are defined synchronization points (“gates”), which terminate certain steps, such as the end of pre-sales negotiations with the customer. By synchronizing at these points, it can be assured that all required steps in the previous phase have been completed and that all necessary information is available to continue. At these synchronization points all affected employees get in contact with the process.

In practice, some concepts from academia have not proved to be useful yet. An example for this is breaking individual services into reusable service components or service modules. It has shown to be quite complicated to identify previous service provisioning projects where a similar solution has been provided. If one takes a firewall as an example, it is often easier for an experienced staff member to customize an off-the-shelf firewall product starting with the default configuration than to search in previous projects whether a similar configuration has been demanded before. An obstacle in using the latter approach is that there is often no suitable kind of directory to search in prior projects for the same or similar configurations. In addition, particularly in individual service provisioning, many previous configurations do not exactly match to the currently given requirements, which requires reconfiguration anyway.

SOAs are not yet relevant in service provisioning for this service provider.

Comments of the second engineer. The solution provider uses mostly home-grown tools for service provisioning for historical and political reasons. For service provisioning, a distinction is made between administrative data and technical data. Administrative data, like contracts, supplier addresses, contact information is contained in SAP, which is cumbersome to extend for other related data, while a home-grown application manages technical data stored in a commercial database.

The ordering of new services is done via a web interface in simpler cases, while additional spreadsheets have to be filled in otherwise. Service provisioning is done according to ITIL processes and employees are responsible for the ITIL manager roles (in contrast to the comments of the first engineer). Nowadays, customers sometimes explicitly demand service providers to be organized according to ITIL standards. A solutions database with previous service provisioning cases is available, but the frequency of actual use is unknown.

Comments of the third engineer. This engineer was involved in a project where ten thousand branch offices of a customer within a country had to be connected to the common Intranet using standard components. Technically speaking, Cisco SOHO (small office/home office) routers were installed at the branch offices and connected via ISDN to the corporate headquarters where an installation was performed to manage the customer connections including remote management of the routers. While the whole service can be seen as an individual service, the provisioning of equal network access for thousands of branch offices shows properties of a commodity service.

For provisioning, a special rollout system was constructed, which was basically a commercial database combined with a self-made script-based workflow system including interfaces to tools like SAP. The rollout database contained workflow descriptions for the steps to be taken for the equipment installation (schedule appointments with branch offices, carry routers to branch offices, install equipment at sites, configure ISDN connection through provider network, finalize equipment installation in accordance with the branch office staff, collect data for accounting and send customer bills, send orders to suppliers).

All branch offices had to be online at a given date, which requires smooth operation not only from the technical point of view. To name an example of an obstacle which had not been taken into account by either project partner, branch office staff did not accept white network cables as this would not match their furniture. Instead of arguing on technical matters, the solutions provider soon found a supplier for network cables colored in different shades of brown and stocked them in their country-wide warehouses.

The rollout system is not disclosed to nor shared with other providers. Internally, it was used for six or more other related projects, being of substantial value in competition.

Challenges for a network service provider include reliable statements on delivery, functionality and availability of network access, such as: Is DSL available at a number of branch office locations? Which bandwidth is supported in each case? When can it be installed?

This large provider uses a cable management system, that runs in three data centers. It provides an automated mapping of locations to possible network connections. The system has capabilities to reserve certain ports in the network, as well as to configure network components. The actual installation is also dependent on employee availability.

In general, vendors offer pre-configuration services for customer premises equipment. The

simplest level of pre-configuration is to configure the address of another configuration server, which is under control of the solutions provider. Configuration services are performed by many individual service providers for large customers including bundling resources from different suppliers.

6.3 Selected Scenarios from Different Articles (Case Studies)

Relatively few service providers disclose information about their principles in provisioning, like on technical matters such as resource selection strategies, resource performance comparison or organizational matters such as processes or organizational models. However, two articles have been found which address some of the goals and challenges mentioned above. As these articles deal with commodity service provisioning they are complementary to the interviews for individual service provisioning.

Provisioning at Google. In [53], Barroso, Dean and Hölzle describe some details about the Google cluster architecture, including strategies on performance evaluation and resource selection, fault handling, and own developments for a special, highly parallel application. The authors name energy efficiency and price per performance ratio as key design factors.

Google's architecture is based on two principles: First, they believe that the best price per performance trade-off for their applications comes from building a reliable computing infrastructure using clusters of unreliable commodity components. Therefore, reliability is provided in software rather than in high-availability hardware, using more than 15 000 unreliable commodity rack servers to build a highly available computing cluster at a low price. Hardware-based load-balancers are used but no redundant power supplies, RAID systems nor costly fault-tolerant components in the servers. Software-based reliability means that services as well as data are replicated across many different machines, with highly automated management systems in place to prevent, detect and work around failures. GoogleFS [54], a file system designed for and custom-built to the special requirements at Google, has been developed and is now in use in the data centers.

Second, the design is tailored for best aggregate request throughput, not peak server response time, since response times can be managed by parallelizing individual requests. The ratio of price per performance is addressed by purchasing the CPU generation which—at the date of procurement—gives the best performance per unit price, not the CPUs that give the best absolute performance. The authors claim that all software is produced in-house, and system management overhead is minimized through extensive automation and monitoring, which makes hardware and energy costs a significant fraction of the total system operating expenses.

Provisioning of web hosting services. Web hosting services are a good example of commodity services, where automation of provisioning is driven to extremes, due to its direct influence on profit. In Germany, two web hosting providers host more than half of the nine million German .de-domains in 2005 [55]. A good description of service provisioning at these providers can be found in [56].

In their web hosting platforms, every provisioning step - from order processing, domain registration to web server configuration - is carried out by thoroughly elaborated and maintained programs written in scripting languages.

Due to the high level of automation, the choice and configuration of resources is crucial to the success of web hosting providers as the following real-world example shows.

- A procured Sun E6500 was one of the biggest servers at a certain period of time, but its strength was more on number crunching than I/O performance.
- A data center was only served by one ISP, resulting in many customers shifting to another provider, when this ISP got into financial trouble.
- Due to budget restrictions, an EMC storage system had not been mirrored to a second one, as had been agreed upon with a contractor.
- Service outages of several days and lost data due to a storage crash caused many customers to transfer their web sites to other hosting providers.

Growing more and more, both companies put substantial efforts in the design of new data centers. Dedicated servers hosted for customers provided a new challenge due to their scale and density, resulting new requirements on available space, energy supply and cooling.

7 Summary

Service provisioning has different faces, depending on the number of orders and the level of customization of the service. The high number of equal and relatively simple orders of commodity services allows automation of many steps in the provisioning workflow. Management systems and scripts are set up for this purpose, but they need constant maintenance and orchestrated resources. In contrast, individual services are characterized by a complex set of specific customer requirements and negotiations between service provider and customer on financial and technical matters. Automation is therefore less applicable. In customized services, service providers combine features of individual and commodity services.

As service provisioning includes allocation and configuration of resources, setting the stage for service operations, it has a direct influence on profit. However, many challenges arise from a number of conflicting goals.

Addressing the organizational aspect of IT service management, frameworks of best-practice type processes such as ITIL and eTOM aim at defining terms and providing guidance on how to structure service provisioning and operations. Standards across service providers such as ISO 20000 allow them to be certified to express a certain level of quality to customers. Based on the terms and processes, vendors can develop service support tools for a wide range of service providers. Service Oriented Architectures play only a minor role in the current practice of service provisioning. However, the concepts are supported by major vendors and can therefore be expected to become increasingly relevant.

As mentioned before, tools, ranging from groupware and project management tools in individual services to flow-through provisioning tools in some commodity services, support many steps in service provisioning. Resource virtualization techniques are already applied in many scenarios, but they are expected to bring even new opportunities in the future. Interoperability among commercial provisioning support tools as well as home-grown software and scripts still leaves many open issues.

Service provisioning research is driven by the increasing complexity in terms of scale and heterogeneity of resources and services as well as their interdependencies. Resource allocation algorithms, which have been studied for a long time, are now further elaborated in context with efforts in modeling of organizations and resources including their structure, behavior, capabilities and dependencies. Physical limitations in terms of cooling large data centers, as

well as the need for increased energy efficiency in fix and mobile networks increasingly influence decisions in service provisioning, posing open research questions.

Research in self-management techniques initiated especially by IBM's Autonomic Computing Initiative aims at encapsulating complexity in autonomic elements instrumented by business goals and policies. Research in process frameworks tackles complexity from the organizational point of view in single- and multi-domain-scenarios. Implementation of basic infrastructures for grid and utility computing has already started, but still a lot of research is needed to make the visions become reality. Ad-hoc and P2P networks question the traditional centralized approach of provisioning resources under control of a service provider, with service providers such as Skype including these mechanisms in their business models.

In daily operations, provisioning is currently based on experienced network and systems engineers, supported by management systems. Future tools promise to take care of most of the operational procedures, but people will be needed to develop, test, run and maintain these tools.

Acknowledgement

The authors wish to thank the members of the Munich Network Management (MNM) Team for helpful discussions and valuable comments on previous versions of this paper. The MNM Team directed by Prof. Dr. Heinz-Gerd Hegering is a group of researchers of the University of Munich, the Munich University of Technology, the University of the Federal Armed Forces Munich, and the Leibniz Supercomputing Center of the Bavarian Academy of Sciences. Its web-server is located at <http://www.mnm-team.org>.

References

- [1] Alexander L. Factor. *Analyzing Application Service Providers*. Sun Microsystems Press, 2002.
- [2] Telemanagement-Forum. *Telecom Operations Map (eTOM) – The Business Process Framework – GB921 (Version 4.0)*, 2004. GB921, Version 4.0.
- [3] Telemanagement-Forum. *Shared Information/Data (SID) Model – Concepts, Principles, and Domains*, 2003. GB922, Version 3.1.
- [4] Telemanagement-Forum. *The NGOSS Technology-Neutral Architecture*, 2004. TMF 053, Version 3.1.
- [5] Telemanagement-Forum. *The NGOSS Lifecycle and Methodology*, 2004. GB927, Version 1.1.
- [6] Telemanagement-Forum. *NGOSS Compliance Testing Strategy – Technical Specification*, 2004. TMF 050, Version 4.1.
- [7] TeleManagement Forum. New Generation Operations System and Software. <http://www.tmforum.org>.
- [8] Telemanagement-Forum. *eTOM – Public B2B Business Operations Map (BOM) Application Note C – An initial proposal for the scope and structure of ICT Business Transactions*, 2004. GB921C, Version 4.0.
- [9] Telemanagement-Forum. *Telecom Operations Map (eTOM) – The Business Process Framework – Addendum D: Process Decompositions and Descriptions*, 2004. GB921D, Version 4.0.
- [10] Telemanagement-Forum. *Telecom Operations Map (eTOM) – The Business Process Framework – Addendum F: Process Flow Examples -*, 2004. GB921F, Version 4.0.
- [11] Telemanagement-Forum. *Telecom Operations Map (eTOM) – The Business Process Framework – eTOM-ITIL Application Note – Using eTOM to model the ITIL Processes*, 2004. GB921L, Version 4.0.
- [12] OGC, editor. *Introduction to ITIL*. The Stationery Office, 2005.
- [13] OGC, editor. *Service Support*. IT Infrastructure Library. The Stationery Office, 2000.
- [14] OGC, editor. *Service Delivery*. IT Infrastructure Library. The Stationery Office, 2001.
- [15] ISO/IEC. *International Standard – ISO/IEC 20000-1:2005 – Information Technology - Service Management - Part 1: Specification*, December 2005.
- [16] ISO/IEC. *International Standard – ISO/IEC 20000-2:2005 – Information Technology - Service Management - Part 2: Code of Practice*, December 2005.
- [17] M. Brenner. Classifying ITIL Processes — A Taxonomy under Tool Support Aspects. In *First IEEE/IFIP Workshop on Business-Driven IT Management (BDIM 06)*. IEEE, April 2006. <http://www.mnm-team.org/pub/Publikationen/bren06/>.

- [18] Architecture of Integrated Information Systems (ARIS). <http://www.ids-scheer.com/aris>.
- [19] Peter F.L. Scheffel and John Strassner. Handbook of network and system administration. chapter IT Service Management. Elsevier, 2007.
- [20] D. Krafzig, K. Banke, and D. Slama. *Enterprise SOA – Service-Oriented Architecture Best Practices*. Pearson, 2005.
- [21] OASIS. *Reference Model for Service Oriented Architecture*, February 2006. Committee Draft 1.0.
- [22] HP OpenView. <http://www.managementsoftware.hp.com/>.
- [23] IBM Tivoli Provisioning Manager. <http://www-306.ibm.com/software/tivoli/products/prov-mgr/>.
- [24] Sun N1 software. <http://www.sun.com/software/n1gridsystem/>.
- [25] Cisco Configuration Engine. <http://cco-rtp-1.cisco.com/en/US/products/sw/netmgtsw/-ps4617/index.html>.
- [26] Mark Burgess. Cfengine: A site configuration engine. In USENIX, editor, *Computing Systems, Summer, 1995.*, volume 8, pages 309–337, pub-USENIX:adr, Summer 1995. USENIX.
- [27] Service Provisioning Markup Language. <http://www.openspml.org/>.
- [28] I. Foster, C. Kesselman, and S. Tuecke. The Open Grid Services Architecture. In *I. Foster and C. Kesselman (eds.): The Grid Blueprint for a new computing infrastructure, 2nd edition*. Morgan Kaufmann, 2004.
- [29] Configuration Description, Deployment, and Lifecycle Management: SmartFrog-Based Language Specification. Technical Report GFD-R-D.051, Global Grid Forum, Sep 2005.
- [30] Configuration Description, Deployment, and Lifecycle Management: Component Model Version 1.0. Technical Report GFD-R-D.065, Global Grid Forum, Mar 2006.
- [31] Configuration Description, Deployment, and Lifecycle Management: CDDLML Deployment API. Technical Report GFD-R-D.069, Global Grid Forum, Mar 2006.
- [32] M. Archarya et al. SOA in the Real World - Experiences. In *Proceedings of the Third International Conference on Service-Oriented Computing (ICSOC 2005)*, pages 437–449, Amsterdam, The Netherlands, December 2005. LNCS 3826.
- [33] OSS through Java Initiative. <http://java.sun.com/products/oss/>.
- [34] A. Clemm, F. Shen, and V. Lee. Generic Provisioning of Services - A Close Encounter with Service Profiles. *Computer Networks, Elsevier*, 43(2003):43 – 57, 2003.
- [35] M. Langer, S. Loidl, and M. Nerb. Customer Service Management: A More Transparent View To Your Subscribed Services. In *Proceedings of the 9th IFIP/IEEE International Workshop on Distributed Systems: Operations & Management (DSOM 98)*, pages 195–206, Newark, DE, USA, October 1998. IFIP/IEEE.

- [36] K. Begnum, M. Burgess, T.M. Jonassen, and S. Fagernes. On the stability of adaptive service level agreements. *eTransactions on Network and System Management*, 2(1):13–21, 2006.
- [37] M. Burgess and K. Begnum. Voluntary cooperation in a pervasive computing environment. *Proceedings of the Nineteenth Systems Administration Conference (LISA XIX) (USENIX Association: Berkeley, CA)*, page 143, 2005.
- [38] Venkatesh Sarangan and Jyh-Cheng Chen. Comparative study of protocols for dynamic service negotiation in the next-generation internet. *IEEE Communications Magazine*, 44(3):151–156, March 2006.
- [39] G. Dreo Rodosek. *A Framework for IT Service Management*. Habilitation, University of Munich (LMU), June 2002.
- [40] A. Keller and R. Badonnel. Automating the Provisioning of Application Services with the BPEL4WS Workflow Language. In *Proceedings of the 15th International Workshop on Distributed Systems: Operations and Management (DSOM 2004)*, Davis, California, USA, November 2004.
- [41] A. Brown, A. Keller, and J. Hellerstein. A model of configuration complexity and its application to a change management system. In *Proceedings of the 9th IFIP/IEEE International Symposium on Integrated Network Management IM'2005*, pages 631–644. IEEE, May 2005.
- [42] Yixin Diao and Alexander Keller. Quantifying the complexity of it service management processes. In Radu State, Sven van der Meer, Declan O’Sullivan, and Tom Pfeifer, editors, *DSOM*, volume 4269 of *Lecture Notes in Computer Science*, pages 61–73. Springer, 2006.
- [43] S. Singhal, S. Graupner, A. Sahai, V. Machiraju, and J. Pryne. Quartermaster: A Resource Utility System. In *Proceedings of the 9th IFIP/IEEE International Conference on Integrated Network Management (IM 2005)*, Nice, France, May 2005. IFIP/IEEE.
- [44] Akhil Sahai, Calton Pu, Gueyoung Jung, Qinyi Wu, Wenchang Yan, and Galen S. Swint. Towards Automated Deployment of Built-to-Order Systems. In *DSOM*, pages 109–120, 2005.
- [45] G. Swint, G. Jung, C. Pu C, and A. Sahai. Automated Staging for Built to Order IT Systems. In *NOMS*, 2006.
- [46] SunTone Service Excellence model. www.suntone.org.
- [47] P. Modi, H. Jung, M. Tambe, W. Shen, and S. Kulkarni. A Dynamic Distributed Constraint Satisfaction Approach to Resource Allocation. In *Principles and Practice of Constraint Programming - CP 2001*, Paphos, Cyprus, November 2001.
- [48] Mark Burgess. An approach to policy based on autonomy and voluntary cooperation. *Lecture Notes on Computer Science*, 3775:97–108, 2005.
- [49] M. Sailer. Towards a Service Management Information Base. In *Proceedings of the IBM PhD Student Symposium at ICSOC 2005*, Amsterdam, The Netherlands, December 2005.

- [50] M. Cochinwala, H.S. Shim, and J.R. Wullert. A Model-Driven Approach to Rapid Service Introduction. In *Proceedings of the 9th IFIP/IEEE International Conference on Integrated Network Management (IM 2005)*, Nice, France, May 2005. IFIP/IEEE.
- [51] T. Eilam, M.H. Kalantar, A.V. Konstantinou, G. Pacifici, J. Pershing, and A. Agrawal. Managing the configuration complexity of distributed applications in internet data centers. *IEEE Communications Magazine*, 44(3):166–177, March 2006.
- [52] S. Graupner, R. Koenig, V. Machiraju, J. Pruyne, A. Sahai, and A. van Moorsel. Impact of Virtualization on Management Systems. Technical Report HPL-2003-125, HP Labs, Palo Alto, 2003.
- [53] L. A. Barroso, J. Dean, and U. Hölzle. Web Search for a Planet: The Google Cluster Architecture. *IEEE Micro*, 23(2):22–28, March/April 2003.
- [54] S. Ghemawat, H. Gobioff, and S. Leung. The Google file system. In *SOSP*, pages 29–43, 2003.
- [55] DENIC eG. Statistics on .de domains, <http://www.denic.de/de/domains/statistiken/>.
- [56] H. Bleich. Die Masse machts – Wie 1&1 und Strato das Webhosting revolutioniert haben. *c’t – Magazin für Computertechnik*, 19(2005):188 – 192, 2005.

All URLs were last checked on May 31st 2006.

Glossary

API—Application Programming Interface

ARIS—Architecture of Integrated Information Systems

ATM—Asynchronous Transfer Mode

B2B—Business to Business

BPEL—Business Process Execution Language

BS15000—British Standard 15000

BSI—British Standards Institution

CAB—Change Advisory Board

CAB-EC—Change Advisory Board Emergency Committee

CBR—Case-Based Reasoning

CCTA—Central Computer and Telecommunications Agency

CEO—Chief Executive Officer

CHAMPS—CHAnge Management with Planning and Scheduling

CIM—Common Information Model

CMDB—Configuration Management Database

CPU—Central Processing Unit

CRM—Customer Relationship Management

CSM—Customer Service Mangement

DNS—Domain Name System

DSLAM—DSL Access Multiplexer

DSL—Digital Subscriber Line

EPC—event-driven process chain

eTOM—enhanced Telecom Operations Map

EXIN—Examination Institute for Information Science

FAB—Fulfillment, Assurance und Billing

FCAPS—Fault, Configuration, Accounting, Performance, Security Management

HTTP—Hypertext Transfer Protocol

ICSP—Internet and Communications Service Provider

ICT—Information and Communication Technology

IEC—International Electrotechnical Commission

ISDN—Integrated Services Digital Network

ISEB—Information Systems Examinations Board

ISO—International Organization for Standardization

ISO/IEC 20000—ISO/IEC 20000: International Standard for IT Service management

ISP—Internet Service Provider

IT—Information Technology

ITIL—IT Infrastructure Library

ITSM—IT Service Management

itSMF—IT Service Management Forum
ITU—International Telecommunication Union

J2EE—Java 2 Enterprise Edition (Sun)

LAN—Local Area Network
LDAP—Lightweight Directory Access Protocol

MIB—Management Information Base
MNMTeam—Munich Network Management Team
MOF—Microsoft Operations Framework
MPLS—Multi Protocol Label Switching

NGOSS—New Generation Operations Systems and Software
NMF—Network Management Forum

OASIS—Organization for the Advancement of Structured Information Standards
OGC—Office of Government Commerce
OGSA—Open Grid Services Architecture
OGSI—Open Grid Services Infrastructure
OLA—Operational Level Agreement
OSS—Operations Support Systems

P2P—Peer to Peer
PIR—Post Implementation Review

QoS—Quality of Service

RfC—Request for Change
RPC—Remote Procedure Call
RSVP—Resource Reservation Protocol

SAN—Storage Area Network
SID—Shared Information/Data (model)
SIP—Session Initiation Protocol
SLA—Service Level Agreement
SLM—Service Level Management
SM—Service Management
SOA—Service Oriented Architecture
SOAP—Simple Object Access Protocol
SOHO—Small Office Home Office
SPML—Service Provisioning Markup Language
SQL—Structured Query Language

TMF—TeleManagement Forum
TNA—Technology Neutral Architecture
TOM—Telecom Operations Map

TPC—Transaction Processing Performance Council
TPC-W—Transactional web e-Commerce benchmark by TPC

UC—Underpinning Contract
UML—Unified Modeling Language

VC—Virtual Circuit
VLAN—Virtual Local Area Network
VPN—Virtual Private Network

W3C—World Wide Web Consortium
WSDL—Web Services Description Language
WS-RF—Web Service Resource Framework
WWW—World Wide Web

XML—Extensible Markup Language

Ethical, Legal and Social Aspects of Systems

Siri Fagernes and Kirsten Ribu

1 Introduction

Communication networks have altered many aspects of life, like work and employment, healthcare, transportation and entertainment. Technology has brought many benefits, but is also the cause of many social and ethical concerns: system failures, the destruction of property, the so called *digital divide*, workplace monitoring, Internet censorship, and the increasing gap between rich and poor, are some examples of important ethical questions that arise when we reflect on how technology has impacted modern life.

System Administrators play an important part in keeping networks secure. They are also responsible for the uninterrupted operation of the computers to take care of the business needs. A system administrator's duties and responsibilities encompass more than technical knowledge. They are given broad access to the resources of computer systems because their job responsibilities require such access. For instance, in troubleshooting email issues, a system administrator may have to go through the employees' email, but must treat any knowledge so gained as private. Many companies have codes of ethics for employees who in the nature of their job have privileged access to company systems. The content of such codes may include statements like: 'do not browse through the computer information of system users while using the powers of privileged access unless such browsing: is a specific part of the job description, do not disclose computer information observed while operating with privileged access, do not copy any computer information for any purpose other than those authorized under their defined job responsibilities, report suspicious incidents to management or police' and so on. Such codes of ethics provide guidelines for professional conduct, and are useful tools. However, situations do arise where it may be hard to decide what to do, for instance, how would you react if your boss asks you to go through all the employees' email?

2 Social Aspects of Technology

New technologies have blossomed during the past century, and have had a tremendous impact on our way of living. Automated methods for developing all kinds of products have lowered the costs and hence the prices, which again has led to increased purchasing power for households. Less manual labour has reduced the physical strain on the human body, which combined with improved methods within the medical profession, has led to increased life expectancy.

However, the overall effects of new technology are never fully predictable, and it is obvious that there are several side effects to this development. While less manual labour has led to less physical strain, various health problems might

emerge through inactivity. While technology has lead to automated production and increased productivity, the people who earlier performed this work manually have become obsolete.

These and other important issues call for increased awareness regarding introduction of new technology, concerning both short-term and long-term effects on our society. In this section we will present several areas that should be studied in more detail.

2.1 Health issues

When discussing the social implications of technology, the ones perhaps raising the biggest concerns are those connected to human health. Increased access to technology has reduced the need for human physical labour, and hence reduced the severe physical strain on the human body as a result of daily work. Also, the expected age has increased dramatically as a result of progress within the medical sciences, assisted by better technologies to aid laboratory analysis, test-taking, speeding up the process for giving diagnosis to illnesses and so on.

However, the dramatic increase in the use of so-called *ICTs* the last decade, has also meant increased exposure to other negative side-effects. Extensive research has been performed on this issue, but it seems very difficult to come to any clear conclusion.

When discussing the health effects of information and communication technology, one distinguishes between the physical effects and the psychological effects. It might be of no surprise that the physical effects of ICT on the human body are a lot simpler to investigate than the psychological effects. The first are also the best documented of the two [1]. Of the physical side-effects, the following are among the most common:

1. *Weight issues.* The extensive use of ICT's, at home, in schools and at work, has lead to less physical activity, and hence to an increase in the number of cases of serious overweight or obesity.
2. *Repetitive strain injuries.* As a result of extensive use of a computer mouse or joystick, severe pain in arms and shoulders has been a common problem for those using a computer as a daily routine. Also, spending long hours sitting in front of a computer screen often lead to extensive strain on the upper body including the back region.
3. *Vision.* The development of the eyes/vision of children could have negative effects of the radiation from TVs and CRT-screens.
4. *Artificial light.* Some types of artificial light (television screens and computer CRTs) might be harmful to biological organisms.
5. *Radiation.* Our daily exposure to electromagnetic signals from several types of networked devices might pose potential risks.

On the psychological side, the results are a bit more difficult to interpret, since the *content* of the media studied could be as important as the amount of consumption [1]. However, there are suggestions of certain physiological effects. Research has shown clear correlations between heavy use of media (television, Internet, video games etc.) and signs of depression and anxiety among young

people [1]. Information overload, and the fact that youngsters are not mentally prepared for all these impressions are listed as causes¹.

Other aspects that could be affected are the general intellectual development, attention span and imagination of users. One particular example is the reliance of external stimulation as a result of heavy consumption of ICT-related media. Also, addiction to certain activities as a result of the new technologies might be a problem in the near future.

2.2 Employment and Work

A common perception is that innovations in technology result in fewer available jobs, because automated production makes human labour redundant. Investigations show that several branches have suffered from reduction in the number of available jobs, both within manufacturing and so-called 'white-collar' jobs [2].

On the other hand, there exist several examples of how innovations within technology have lead to an increase in the number of available jobs, because new technology creates a need for new job functions. Also, innovations cause prices of different goods to drop, again leading to increased purchasing power among people in general, which again leads to increased demand for the products, and hence more jobs.

Teleworking, as a result of improved communication technology, has resulted in improved working conditions for those dependent on working from home. This makes it possible for single parents without means to pay for child care, people living in rural areas, and others who normally would have difficulties working normal office hours, to take part in the job market. However, common experiences are a lack of contact with colleagues and office environment, difficulties in separating office hours and leisure time, since office and home are at the same location, and sometimes missed opportunities in getting promoted.

2.3 Changing Social and Group-dynamical Patterns

One of the most influential technological changes during the last decade has been the mobile phone. Teenagers have adopted this new technology and let it change the way they interact and keep in touch, even using it as a basis for organising their circle of friends. This is an example of how *not* having a cellular phone could cause one to be excluded from the social community, as it is the main source of communication and exchange of information.

In the Internet community, there are examples of office environments changing their meeting habits, now using chatrooms as a forum for office meetings. When meetings and general discussions take place 'online' instead of in a physical meeting room, changes in group dynamics emerge, such as who dominates the group, as there typically are different skills needed to succeed online. This is an example of how new technology might fundamentally change the way humans interact, and affect which groups that benefit from the interaction form. While typical 'nerds' might be intimidated by physical meetings, they can 'rule' online.

New technologies also have an impact on the family and home environment. Teenagers tend to adopt new technology more easily than their parents, and

¹Editorial comment: Omega-3 deficiency has been related to youth apathy and media addiction in trials in the U.K. indicating that media dependency is part of a wider problem.

hence gain more 'power' in the home, as they become the 'gurus' of the family. This has several implications for the dynamics between parents and their children. For instance, how can the parents control the Internet user habits of their children if the children are in charge of the computers in the home. This phenomenon disturbs the normal family pattern where the parents are the role models.

2.4 The Digital Divide

As information technology has gained foothold in our society, a phenomenon known as the *digital divide* has emerged. A common definition of the digital divide is *the situation where some people have access to modern information technology while others do not*. [2]. Usually one speaks of the *global* and the *social* divide, where the global divide is a measure of the difference in Internet access between developed and less developed countries, while the social divide is a reference to the difference between individuals within the same country [2]. Parameters affecting the level of Internet access within a particular country are, among others, race, income level and age group.

The global divide is illustrated by the numbers of people in the world with Internet access. In June 2006, approximately 1,043 million people worldwide use the Internet, according to Internet World Stats. The United States, Australia and Europe are well above the average, while other parts of the world are below. In Africa, only about 1 out of 200 persons had Internet access in 2000 (2.6 per cent). Because of the high numbers of Internet users in the West, one tends to forget that most people in the world actually have little access to not only the Internet, but also telephones, health care, education and books. These problems are caused by poverty, isolation and politics. Also, there is a digital divide in the United States and other western countries between the rich and the poor, people of different age groups, and between people of different ethnical background. The consequences are serious: people who have not acquired basic computer skills will have trouble finding jobs, and will have little chance to escape poverty. Thus, the global digital divide will continue to exist.

When discussing the digital divide, one important concept is *technological diffusion*, which refers to *the rate at which a new technology is assimilated into a society*. There exist two common theories on how this happens:

1. *The normalization model*. This model predicts that *everyone* in a society will eventually adopt to a new technology, but that the wealthiest part of the population adopts first, then the middle range, and finally the least wealthy part. This model is referred to as the 'optimistic' model.
2. *The stratification model*. This model predicts that the income groups will adopt in the same order as the other model, but it contradicts in the sense that it claims that a larger percentage of the wealthier groups will adopt than the less wealthier. This means that a certain part of the population will *never* take to the new technology, and describes a pessimistic model.

The reason why the digital divide is such a concern nowadays, is the relationship between 'digital literacy' and general wealth or success. Some of the criticism against the digital divide lies in the fact that it appears that digital literacy follows from general success, and not the other way round.

3 Ethics

3.1 Ethics and Morality

The term 'ethics' defines the philosophical study of morality, whereas the term 'morality' refers to the way we live and act. To simplify this, we can say that if the traffic laws and regulations are the underlying ethics of driving, morality is abiding by these laws and regulations. Speeding and drunk driving is obviously immoral conduct. Some behaviour is not regulated by law, but rather agreed on, for instance polite and considerate driving. Traffic bullies are therefore immoral. In this way we see that ethics reach beyond the law, and philosophers have at all times occupied themselves a great deal with the theoretical study of ethics, or moral philosophy. There are many different perspectives on ethics, as we shall see.

Ethical dilemmas arise when we are faced with problems of right or wrong. The difference between right and wrong may be hard to determine, and is not necessarily a question of what is legal. Tying 'good' and 'bad' to expressions of approval or disapproval of human actions may be difficult, because things can be good or bad independently of our personal approval or disapproval of them [3].

Certain ethical judgements can be classified as feelings or emotions, and these are neither true or false. This view is known as emotivism. However, most philosophers refer to our actions and intentions when discussing moral matters, not to our emotions, because this belongs more to the field of Psychology. The challenge of showing that people are better off acting morally than immorally may be difficult. Personal gain and individual convictions play an important part in most peoples lives. Although we know that copying proprietary software or downloading music may be illegal, many people are opposed to the laws regulating such activities, and defend their actions by attacking unpopular regulations. So in order to decide whether something is good or bad, right or wrong, one must rely on experience and common sense, and judge by the consequences or intentions of the act. A knowledge of ethical principles may be a help when trying to analyze difficult ethical dilemmas.

Changes in technology require changes in laws and personal attitudes. The use of computers can cause new ethical dilemmas because people are pushed into unforeseen situations. Although many questions that arise from the use of computers are not very different from problems encountered in other aspects of life, the activities on the Internet, the sheer size of the World Wide Web and the amount of available information often create new ethical dilemmas. Computers are a part of our everyday life, both professional and private, and we are dependent on computer networks in all parts of society. A problem with a computer system may negatively impact a whole company, the company's business partners, financial interests as well as hundreds of users. The interest in ethical reflection on the use of computer systems is steadily growing, and the study of ethics is incorporated into the computer science curricula in many universities and colleges.

The growth of the Internet has created several new legal issues. Boundaries are no longer geographical, and this fact can be used to circumvent national laws. For instance, Internet-based casinos are illegal in many countries, therefore, gambling web-sites are run from places like the Caribbean or Malta. Some of

the main concerns in computer ethics today are questions of copyright, privacy and censorship. Traditional rules of conduct are not always applicable to a new medium. It is necessary to reflect on questions like the following: If a certain nation wishes to protect freedom of speech on the Internet, whose laws apply? What about censorship on the Internet? A national constitution protecting freedom of speech is in a sense a local law which does not necessarily apply in other countries. How can issues like freedom of speech, protection of intellectual property, invasions of privacy be governed by law when the whole world is involved?

It is important to become aware of the fact that there are not necessarily correct answers to ethical dilemmas. Every computer professional should therefore have some knowledge of international law and best practices, and an awareness of the responsibilities tied to the job, and be acquainted with the codes of ethics that exist for members of the profession. He or she should reflect on how to deal with ethical dilemmas in situations where there may be considerable pressure to do something illegal or ethically wrong.

3.2 What are *Computer Ethics*?

Ethics and values are today discussed in a context that is not limited to a particular geographic region, a specific religion or culture. Much of the discussion on ethics today has to do with the use of modern technology. Computers are everywhere, playing a large part of peoples' lives, both professionally and in the field of entertainment and relaxation. Computer ethics can therefore be understood as that branch of applied ethics which studies and analyzes social and ethical impacts of information technology.

Computer ethics involve the use of computers, networks and the Internet, and therefore also include global information ethics. Computers have become commonplace, and have great impact on society and the transformation of culture. Because computers have become universal and are part of our daily lives, many ethical problems have to do with computer ethics. Although problems involving computing connect with every-day ethical issues in some way, the field of computer ethics is also unique because of the technology itself. Computers can be designed to perform any task we wish. They are universal tools that can be used in so many ways that it is difficult to foresee all the situations that may arise. Therefore, we do not always have traditions or policies to act as ethical guidelines.

Computers tend to create more novel situations and dilemmas than other tool and technologies. The technology provides opportunities for misuse. For instance, newspapers commonly report stories about identity theft on the Internet, body parts being sold, violation of copyright and privacy, illegal monitoring, terrorists and criminals using the Internet for their own ends, and crimes like the spreading of child pornography.

A great deal of literature on computer ethics discusses the laws and regulations concerning such issues and the use and misuse of the Internet, freedom of speech, privacy, intellectual property, and professional conduct. However, ethics reach beyond the law. Modern technology is developing at a tremendous speed, and traditional ethical theory and every day morals do not always help us to make the right decisions when dealing with computer technology. A system administrator comes into contact with privileged information and must protect

the confidentiality of all such information. However, a system administrator may be pressured into exposing such information, for instance snooping in employees' e-mail. The knowledge of ethical principles may serve as a help in order to decide how to act morally under pressure.

3.3 Ethical Principles

Moral truths are those truths that are accepted by many people, despite differences in culture and religion. One example is Human rights. Others could be deeds that create peace and harmony between people. Ethical rules like '*do not steal or cheat, keep promises, pay your debts and do not lie*' ensure that we live good lives in interaction with other people. So how do we decide what is good, fair and honest, in short, how do we behave ethically? One way of looking at it is to place focus on what we ought to do, not what we want to do, which means acting in a certain way according to moral rules. For instance, in a professional context, doing good ethically means doing a good job. On the other hand, good qualities do not necessarily lead to good deeds. Intelligence and courage are good qualities, but can be used in harmful way, for instance for robbing banks, fraud etc.

A useful ethical theory makes it possible for us to examine moral problems, reach conclusions through logical reasoning and defend the conclusions. Philosophers have always discussed moral matters. Formal study of ethics goes back to the greek philosopher Socrates. Philosophers have proposed many ethical theories. Traditionally, there are several moral perspectives, for instance ethical principles of consequence (utilitarianism), duty and virtue, formulated by the philosophers John Stuart Mill, Immanuel Kant and Aristotle respectively. A few words about these moral perspectives will help to understand different views on ethics, and show that as individuals, people think differently about moral values. One should regard these theories as tools, or as the 'operating system' for our behavior as human beings. The theories exist as moral perspectives underlying our actions, but are not visible in our every-day lives.

There are three main categories of practical ethical theories, or normative ethics. *Normative ethics* is the philosophy of practical moral standards that tell us what is right and what is wrong, and how to live moral lives. They focus on the virtues and good habits that we should acquire, the duties that we should follow, or the consequences of our behavior. These three categories are the ethics duty (deontology), consequence, and virtue.

The *theory of duty* (deontology) focuses on the study of right and wrong, obligation, permissions and duty. Theories of duty propose standards of morality, moral codes and moral rules. One rule that is well known is the so called golden rule: 'Do unto others as you would have them do unto you'. This rule focuses on good deeds. Many companies today have moral codes for company behaviour based on this principle.

Another way of looking at ethics of duty it is to consider what it means to do good by acting according to good will. Good will is the only thing that can be called good without qualification. Is not the same as good deeds, but good in itself. Doing the right thing with no thought of the consequences is an act of duty and good will.

Utilitarianism or the *ethics of consequence* (from the Latin utilis, useful) is a theory of ethics based on quantitative maximization of good for society or

humanity. This good can be viewed as happiness or pleasure. Utilitarianism is a form of consequentialism, meaning that an act should be judged by the consequences of that act. Utilitarianism is sometimes summarized as '*The greatest happiness for the greatest number of people.*' Sometimes people say: 'If I had known the consequences, I would not have acted in this way'. Knowing what is good in a consequentialistic perspective may become more clear in hindsight.

Virtue ethics may be identified as the one that emphasizes the virtues, or moral character of a person. In contrast to the approach which emphasizes duties or rules. In philosophy, the phrase *virtue ethics* refers to ethical systems that focus primarily on what sort of person one should try to be. According to virtue ethicists the aim of all humans is to lead a good, happy and fulfilling life.

An ethical principle is a tool for considering whether an act is morally right or wrong. In order to illustrate the difference in the three perspectives described above, we can say that in the case of rescuing a drowning person, a *utilitarian* will point to the fact that the consequences of jumping into the water and pulling the person out will maximise everybody's well-being, the rescuer's own included, since good deeds are rewarded, a *duty ethicist* will state that a rescue act is in accordance with the moral rule '*do unto others as you would be done by*', and not think of any personal gain, whereas a *virtue ethicist* simply knows that rescuing the person is a charitable act, and will do so accordingly.

4 Professional Ethics

Professional ethics differ from general ethics. The professional is an expert in a field that the customer knows little about, for instance computer science or medicine. Customers rely on the knowledge, expertise and honesty of the professional. The products (medicine, computer networks, bridges, investment advice) affect large numbers of people. A computer professional's work can affect the life, health, finances, freedom and future of a client or a group of people. A professional can cause great harm through dishonesty, carelessness or incompetence. People are often unable to protect themselves, not being involved in the choice of tool or product. Therefore, computer professionals have responsibilities to the general public.

Professional Ethics concern one's conduct of behaviour and practice when carrying out professional work. Any code may be considered to be a formalisation of experience into a set of rules. A code is adopted by a community because its members accept the rules, including the restrictions that apply. Apart from codes of ethics, professional ethics also concern matters such as professional indemnity. No two codes of ethics are identical. They vary by cultural group, by profession and by discipline.

It is often assumed that universal ethical principles exist. However, it is a fact that in some cultures, certain behaviours are unacceptable, but in other cultures the opposite may be true, for instance when it comes to the question of software piracy, several viewpoints exist. Richard Stallman of the Free Software Foundation[4] for instance has argued at length about the superiority of freedom over protectionism in software licensing.

In many situations, there are several options that may be ethically acceptable. It is therefore misleading to divide all acts into two categories: ethically right and ethically wrong. One should rather view acts as ethically obligatory,

prohibited or acceptable. One way to try to define these categories are by creating codes of ethics.

After episodes like the Enron scandal of 2001, focus is increasingly placed on ethical behaviour in business². The Enron Corporation was an American energy company based in Houston, Texas, United States. Enron employed around 21,000 people and was one of the world's leading electricity, natural gas, pulp and paper, and communications companies. It went bankrupt in 2001, when it was revealed that its reported financial condition was sustained mostly by systematic accounting fraud. It was the biggest bankruptcy in US history, costing 4,000 employees their jobs. The scandal has become a popular symbol of willful corporate fraud and corruption[5].

Many large companies today are being pressured into acting morally, and are forced to take ethics seriously as a part of their culture. There is a heightened awareness of ethical issues in many societies. Attitudes change, and hopefully, corporate businesses will place ethics higher than profit. Corruption can then be more easily revealed, and experience will show that good ethics will pay in the long run. One reason is that customers are more likely to trade with a business known to be honest and trustworthy. Also, employees will join and faithfully serve a company that treats its workers with loyalty and respect.

Although assuming that doing the right thing pays, it is still not easy to know the best way to balance duties to shareholders, the rights of employees, service to customers and obligations to society. But a good ethical policy can help develop a more honest corporate culture.

According to Deloitte and Touche, *'companies that follow both the letter and the spirit of the law by taking a "values-based" approach to ethics and compliance will have a distinct advantage in the marketplace.'* Deloitte has suggested guidelines for writing a code of ethics, emphasising how an organization should develop its own code to suit the needs of *'its personnel in defining expected behaviors and in addressing the risks, challenges, and customs in the countries in which it operates, as well as to fit their specific industry and situation.'* [6]. The code should not be written as a list of 'do not's' but state expected behaviour, and be global in scope. This example demonstrates that serious attempts are being made in the world of business to behave ethically, although there are, unfortunately, numerous examples of the opposite. However, companies risk having their unethical activities exposed by whistleblowers, who are within their rights to do so, and may be protected by whistleblower protection acts.

4.1 Ethical and Professional Conduct in System Administration

It is useful to have general guidelines for professional and ethical conduct like the codes of ethics referred to above, but ethical dilemmas arise in unexpected situations, and as a systems administrator you may be at loss what to do when for instance you are asked by a colleague or your boss to read someone else's email, or you come across information that reveals plans for sabotaging the company, stealing equipment and reselling via online auction, or illegal financial activities in your company. There are numerous such examples, and you may

²Editorial comment: the Sarbanes-Oxley (SOX) act was introduced as a measure to avoid future debacles in the US.

be faced with a choice: Either 'blow the whistle' and risk losing your job, or keep the information to yourself, having to live with the knowledge that you have maybe not acted ethically.

Regrettably, there are no simple answers to these dilemmas. Moral philosophy assumes that people are rational and make free choices since free choice and the use of rational judgement are characteristics of human beings. Ethical principles may help us out in difficult situations. However, it is human to make mistakes. Besides, a person may not be making a free choice when he or she risks losing a job. Difficult dilemmas arise in professional life, for instance, admitting to a customer that your software is faulty, declining a job for which you are not qualified or speaking out when you know something is wrong takes courage, and is not always rewarded.

The role of the System Administrator

A system administrator's job is to ensure proper operation, support, and protection of a company's network services and to take all security measures to protect the computers and the data contained on them. System administrators have access to an enormous amount of information, and therefore have great power and responsibility. This responsibility also includes abiding by copyright laws, as companies caught using pirated software have significant legal and financial liabilities. Protecting the integrity of information includes ensuring that unauthorized users access or make changes to data not belonging to them. System administrators are often blamed for copyright violations found on their networks.

Generally, smaller companies employ one or two system administrators to look after the organisation. The scenario is different in larger organisations, where there is a larger number of system administrators, each responsible for a specific role within the system.

Unlike many professions, there is no single path to becoming a system administrator. In some cases, candidates are required to possess industry certifications before being considered, but since few colleges or universities have specific programs for system administration, many systems administrators are self-taught. This means that a system administrator may be lacking in knowledge of legal matters and ethical and professional conduct. We therefore seek to give an overview of some important topics that may be useful in order to become aware of ethical questions and how to solve dilemmas. We present references to international laws and regulations, ethical codes and online help-centers. It is our aim to enhance reflection and conscious decision-making.

4.2 Codes of Ethics

Codes of Ethics are concerned with a range of issues, including academic honesty, adherence to confidentiality agreements, data privacy, handling of human subjects, impartiality in data analysis and professional consulting, professional accountability resolution of conflicts of interest and software piracy.

Many computer professionals are members of organisations like the ACM or the IEEE, and so have agreed to be bound by one of the following:

- The ACM Code of Ethics and Professional Conduct

- The ACM/IEEE Software Engineering Code of Ethics and Professional Practice
- The IEEE Code of Ethics

These codes are rather general in their phrasing. They present lists of do's and don'ts, which are easy to agree with, but do not necessarily help us out when we are faced with serious ethical dilemmas. The ACM Code includes general moral imperatives, such as '*avoid harm to others*' and '*be honest and trustworthy*'. The code also included more specific professional responsibilities like '*acquire and maintain professional competence*' and '*know and respect existing laws pertaining to professional work.*' [6]. The IEEE Code of Ethics includes such principles as '*avoid real or perceived conflicts of interest whenever possible*' and '*be honest and realistic in stating claims or estimates based on available data.*' [7].

The Accreditation Board for Engineering Technologies (ABET) has long required an ethics component in the computer engineering curriculum. And in 1991, the Computer Sciences Accreditation Commission/Computer Sciences Accreditation Board (CSAC/CSAB) also adopted the requirement that a significant component of computer ethics be included in any computer sciences degree granting program that is nationally accredited. Professional organizations in computer science recognize and insist upon standards of professional responsibility for their members.

The ACM Software Engineering Code of Ethics and Professional Practice

The Association for Computing Machinery (ACM) has created a code of professional ethics for software professionals. '*Commitment to ethical professional conduct is expected of every member (voting members, associate members, and student members) of the Association for Computing Machinery (ACM).*' This Code, found at the URL <http://www.acm.org/constitution/code.html>, '*consists of 24 imperatives formulated as statements of personal responsibility, and identifies the elements of such a commitment. It contains many, but not all, issues professionals are likely to face....It is understood that some words and phrases in a code of ethics are subject to varying interpretations, and that any ethical principle may conflict with other ethical principles in specific situations. Questions related to ethical conflicts can best be answered by thoughtful consideration of fundamental principles, rather than reliance on detailed regulations.*' [7].

The code says:

'As an ACM member I will . . .

- 1.1** Contribute to society and human well-being
- 1.2** Avoid harm to others.
- 1.3** Be honest and trustworthy
- 1.4** Be fair and take action not to discriminate
- 1.5** Honor property rights including copyrights and patents
- 1.6** Give proper credit for intellectual property
- 1.7** Respect the privacy of others
- 1.8** Honor Confidentiality

ACM/IEEE-CS Joint Task Force on Software Engineering Ethics and Professional Practices

Software engineers shall commit themselves to making the analysis, specification, design, development, testing and maintenance of software a beneficial and respected profession. In accordance with their commitment to the health, safety and welfare of the public, software engineers shall adhere to the following 'Eight Principles:'

PUBLIC - Software engineers shall act consistently with the public interest.

CLIENT AND EMPLOYER - Software engineers shall act in a manner that is in the best interests of their client and employer consistent with the public interest.

PRODUCT - Software engineers shall ensure that their products and related modifications meet the highest professional standards possible.

JUDGMENT - Software engineers shall maintain integrity and independence in their professional judgment.

MANAGEMENT - Software engineering managers and leaders shall subscribe to and promote an ethical approach to the management of software development and maintenance.

PROFESSION - Software engineers shall advance the integrity and reputation of the profession consistent with the public interest.

COLLEAGUES - Software engineers shall be fair to and supportive of their colleagues.

SELF - Software engineers shall participate in lifelong learning regarding the practice of their profession and shall promote an ethical approach to the practice of the profession. See [7].

IEEE Code of Ethics

The Institute of Electrical and Electronics Engineers, IEEE, also has a code of ethics, which is somewhat more explicit. It is found in [8].

'We, the members of the IEEE, in recognition of the importance of our technologies in affecting the quality of life throughout the world, and in accepting a personal obligation to our profession, its members and the communities we serve, do hereby commit ourselves to the highest ethical and professional conduct and agree:

1. To accept responsibility in making engineering decisions consistent with the safety, health and welfare of the public, and to disclose promptly factors that might endanger the public or the environment;
2. To avoid real or perceived conflicts of interest whenever possible, and to disclose them to affected parties when they do exist;

3. To be honest and realistic in stating claims or estimates based on available data;
4. To reject bribery in all its forms;
5. To improve the understanding of technology, its appropriate application, and potential consequences;
6. To maintain and improve our technical competence and to undertake technological tasks for others only if qualified by training or experience, or after full disclosure of pertinent limitations;
7. To seek, accept, and offer honest criticism of technical work, to acknowledge and correct errors, and to credit properly the contributions of others;
8. To treat fairly all persons regardless of such factors as race, religion, gender, disability, age, or national origin;
9. To avoid injuring others, their property, reputation, or employment by false or malicious action;
10. To assist colleagues and co-workers in their professional development and to support them in following this code of ethics.'

A copy of the code can be downloaded at the IEEE web-site.

SAGE

The System Administrators Guild, SAGE, has its own code of ethics: The System Administrators' Code of Ethics.

The SAGE 'vow' is the following: *We as professional System Administrators do hereby commit ourselves to the highest standards of ethical and professional conduct, and agree to be guided by this code of ethics, and encourage every System Administrator to do the same.*

The code is found at [9], and contains advice to the system administrator in the form of a vow, for instance, 'I will be honest in my professional dealings, and forthcoming about my competence and the impact of my mistakes.' The code deals with professionalism, personal integrity, laws and policies, education, and professional and ethical responsibilities. It is interesting to note that the code places emphasis on self-education, especially concerning laws, regulations and policies, as well as the responsibility for updating technical knowledge and work-related skills. System administrators often lack formal education, and the code makes it clear that every system administrator is personally responsible for acquiring appropriate knowledge of technical, legal and moral matters.

The Online Ethics Center for Engineering and Science

The online ethics center [10] contains advice on such topics as engineering practice, responsible research, computers and software, diverse workplace, and gives examples of engineers and scientists who have behaved wisely in difficult situations. The Center's mission is to *provide engineers, scientists, and science and engineering students with resources for understanding and addressing ethically*

significant problems that arise in their work, and to serve those who are promoting learning and advancing the understanding of responsible research and practice in science and engineering [11].

The online Ethics also gives guidelines for citation of web-pages, in order to avoid plagiarism: *The American Psychological Association's recommendations for citing electronic sources, according to the 5th edition of the APA Publication Manual (2001). Citation of the entire website rather than a specific document requires only the address of the site. (i.e. <http://onlineethics.org>) Citation of a specific document requires the author of the page (if listed), the title of the page, the year the page was created or last modified (found below the left-hand menubar on the graphics version of each page on our site and on the third line of the text version), the title of the website, the date you accessed the page, and the URL.*

MLA Guidelines for avoiding plagiarism are found at [10].

The Modern Language Association recommends the following format for citing individual pages in professional sites on the web. A proper MLA citation should at the least include the author of the page (if listed), the title of the page, the title of the website, the date the page was created or last modified (found below the left-hand menubar on the graphics version of each page on our site and on the third line of the text version), the sponsoring organization, the date you accessed the page, and the URL.

Ethics Help-line

The Ethics Help-Line is intended to provide advice for engineers, scientists, and trainees encountering ethical problems in their work. A principal goal of this Help-Line is to assist scientists and engineers in maintaining high ethical standards and in acting wisely when confronted with multiple and potentially conflicting responsibilities, even where this may lead to conflicts with organizational superiors.

The Ethics Help-Line is sponsored by the Online Ethics Center for Engineering and Science and is cosponsored by the Institute of Electrical and Electronics Engineers (IEEE), and the National Institute for Engineering Ethics (NIEE)[12].

A section of the online ethics centre deals with computers and software. *'This section of the OEC contains cases, discussions, and ethical guidelines bearing on the professional responsibilities of computer scientists, computer engineers, and software designers and engineers. These range over fields such as computer theory, computer architecture, and systems engineering.'*

One of the scenarios presented explores the ethical issues involved in the electronic monitoring of the secretarial staff's email[13].

4.3 Whistleblowing

If you discover unethical or illegal actions at work like a violation of law, rule, regulation or a direct threat to public interest, fraud, health, safety violations, and corruption, you must make a decision about what to do with this information. *Whistleblowing* is the term used to define an employee's decision to disclose this information to an authority, for instance your boss, the media or a government official.

Whistleblowing takes moral courage. It means acting according to values of right and wrong when dominant values decree otherwise, being brave enough to stand up to people who have power, considering and understanding the possible consequences and still making the choice to take the action. Despite the dilemmas that potential whistleblowers face, there are increasing numbers of people who are prepared to question corrupt or negligent acts in the workplace, challenging authority by speaking out, even where it may mean risking jobs, career or safety to do so.

Whistleblower Rights

Whistleblowers have rights. Courts have awarded whistleblowers their actual wage loss until they find another job, the difference between their wages for a reasonable time in the future, lost benefits, reinstatement in the previous job, attorney's fees, and punishment, damages. The major question is whether and how to blow the whistle. The Public Interest Disclosure Act 1998 protects workers who 'blow the whistle' about wrong doing. It applies where a worker has a reasonable belief that their disclosure tends to show one or more of the following offences or breaches:

- A criminal offence.
- The breach of a legal obligation.
- A miscarriage of justice.
- A danger to the health and safety of any individual.
- Damage to the environment; or
- Deliberately covering up of information tending to show any of the above.

Whistleblower protection laws are intended to make it safe for employees to disclose company misconduct. Legal protection for whistleblowing varies from country to country. In the US, the Office of Special Counsel enforces the whistleblower protection provision of the Civil Service Reform Act of 1978, Pub.L. No. 95-454, 92 Stat. 1111 (1979), as amended by the Whistleblower Protection Act (WPA) of 1989. The WPA makes it illegal to take or threaten to take a *personnel action* against a federal employee because the employee has made a protected disclosure. *Personnel action* is broadly defined to include virtually any employment-related decision that has an impact on an employee at the worksite.

In the UK, the Public Interest Disclosure Act 1998 (PIDA)[14]:

provides a framework of legal protection for individuals who disclose information about matters of malpractice, and protects whistleblowers from victimisation and dismissal. Under PIDA, whistleblowers must use prescribed channels for making disclosures in order to retain the Act's protection. The Act's preference is that the disclosure be made to the employer itself or an appropriate public authority, rather than the media. In this manner, the Act protects an employee if he or she makes a disclosure in good faith to the employer.

It is not always easy to determine when whistleblowing may be the right thing to do. Informing on illegal or unethical practices in the workplace is difficult for several reasons. Ethical justifications for whistleblowing are frequently uncertain. An important question to answer is: when is it in the public interest to do so? Whistleblowers should be quite certain of their case before they decide to blow the whistle.

The Cases of Inez Austin and Per Yngve Monsen

Although whistle-blowers are protected by law, they may experience great personal strain and professional problems, with the loss of work and reputation as a result. They may become ostracized and unpopular.

The online Ethics Center for Engineers and Scientists contains several examples of moral conduct. One is the case of the whistleblower Inez Austin, who received the Scientific Freedom and Responsibility Award from the American Association for the Advancement of Science (AAAS) *for her courageous and persistent efforts to prevent potential safety hazards involving nuclear waste contamination* as a senior engineer at the Hanford Site in Washington state, a 586-square-mile former plutonium production facility. According to the AAAS,

her stand in the face of harassment and intimidation reflects the paramount professional duty of engineers to protect the public's health and safety and has served as an inspiration to her co-workers.

The Online Ethics Center presents the case of Inez Austin as an example of someone who followed her ethical convictions in the face of overwhelming adversity and refused to sanction a procedure she believed to be unsafe. In June 1990, Inez Austin refused to approve a plan to pump radioactive waste from an aging underground single-shell tank at the Hanford Site to a double-shell tank, because she believed that the process was too dangerous. By blowing the whistle, her life changed, although she came to be regarded highly by environmental and ethics groups, she was subjected to a career-destroying combination of harassment, bureaucratic maneuvering, and ostracization. A second whistleblowing incident a few years later led to the loss of her job[11, 15].

This is the dilemma that a whistleblower must consider. Whistleblowers may be faced with harassment and legal pursuit. Although they receive support from media and government, they may later have difficulties finding a job, for who will hire a 'troublemaker?' In all probability, many companies have something to hide, and a well-known whistle-blower is seen as a risk.

The case of Per Yngve Monsen, the Norwegian employee at Siemens who blew the whistle on the company for corruption and financial fraud, demonstrates the hazards a whistleblower may be exposed to. He was sacked, and even though he won the court case, has been on sick-leave and out of work for nearly two years. In the media he became a hero, and like Inez Austin, an example of good moral conduct. But from his experience with a powerful company, he advises whistle-blowers to be careful, and give great consideration to the risks involved. 'If I had known what I know today, I would never have acted like I did', is his conclusion, reported by the Norwegian newspaper Aftenposten. The same newspaper reported on November the 6th, 2006, that inquiries show that people on the whole are afraid to report illegal or unethical activities because of the consequences.

This is indeed unfortunate, and demonstrates in full that when the individual is up against a powerful international company, the results can be disastrous for the person involved. If you do discover illegal or unethical behaviour, do not hesitate to get in touch with whistle-blower protection organisations and receive legal advice before you blow the whistle.

5 Cyber Law

The phrase *Cyber law* usually refers to rules and laws governing the use of communications technology and informations systems. The big challenge regarding this topic is the fact that the Internet spans on a world wide basis, while laws typically varies from country to country. Also, laws regulating our physical society, are not necessarily sufficient to cover all important aspects of cyber space.

Issues covered by cyber law are

- *Intellectual Property*
- *Privacy*
- *Freedom of expression*
- *Jurisdiction*

5.1 Intellectual Property

Intellectual property denotes the specific legal rights which authors, inventors and other IP holders may hold and exercise, and not the intellectual work itself[16].

Intellectual Property is a legal entitlement which sometimes attaches to the expressed form of an idea, or to some other intangible subject matter.

The motivation of Intellectual Property is to protect inventors, authors etc by giving them the *ownership* of the result of their brain, similarly to the way people own their physical property. The term intellectual property reflects the idea that this subject matter is the product of the mind of the intellect, and that intellectual property rights may be protected by law in the same way as any other form of property.

Electronic media and the WWW creates new challenges for the protection of intellectual property, and new controversies about how intellectual property law. Some of the problems posed by the technology itself are storage and copying of information like text, sound and graphics, and the compressed formats like MP3, which makes copying and distribution of music simple. Downloading and copying text material from the Internet is simple, and makes cheating, for instance, in schools and universities easy. Peer-to-peer technology permits easy transfer of files over the Internet, affecting typically the music and film industry[17].

Digital Rights Management (DRM) is a technology used by copyright owners to control access to digital data. The term often is confused with copy protection, however, DRM restrict the actual use of digital content by way of installing the technology as part of the design. Today, illegal copying and distribution of copied material can result in severe penalties.

Digital Rights Management is a controversial topic. Copyright holders claim that it is their right to restrict copying by the use of technology, because copyright is not respected. On the other hand, the Free Software Foundation is a severe critic of DRM, stating that the word "Rights" is misleading, and that the term 'Digital Restrictions Management' would be more appropriate. The Electronic Frontier Foundation considers some DRM schemes to be anti-competitive.

We have four main categories of intellectual property; copyright, patents, trade marks, and designs.

5.2 Copyright

Copyright is a protection that covers published and unpublished literary, scientific and artistic works, whatever the form of expression, provided such works are fixed in a tangible or material form. Copyright laws grant the creator the exclusive right to reproduce, prepare derivative works, distribute, perform and display the work publicly. Copyright protects the actual piece of work, not the ideas, knowledge or methods used for creating that work.

Copyright gives the creators of a wide range of material, such as literature, art, music, sound recordings, films and broadcasts, economic rights enabling them to control use of their material in a number of ways, such as by making copies, issuing copies to the public, performing in public, broadcasting and use on-line. It also gives moral rights to be identified as the creator of certain kinds of material, and to object to distortion or mutilation of it [16]. Copyright laws enable those who create a piece of work to decide if and how the work can be used, reproduced etc. It is possible to let others use the work freely, or use and distribution must be paid for. There is no such thing as an International Copyright. Protection against unauthorized use in a particular country depends on the national laws of that country. However, most countries do offer protection to foreign works under certain conditions, and these conditions have been greatly simplified by international copyright treaties and conventions. For instance, the Berne Convention for the Protection of Literary and Artistic Works defines the nature and use of intellectual property[18].

Article 2 in the convention defines Protected Works as:

- Literary and artistic works
- Possible requirement of fixation
- Derivative works
- Official texts
- Collections
- Works of applied art and industrial designs
- News

According to the convention, *the expression 'literary and artistic works' shall include every production in the literary, scientific and artistic domain, whatever may be the mode or form of its expression, such as books, pamphlets and other writings; lectures, addresses, sermons and other works of the same nature; dramatic or dramatico-musical works; choreographic works and entertainments in*

dumb show; musical compositions with or without words; cinematographic works to which are assimilated works expressed by a process analogous to cinematography; works of drawing, painting, architecture, sculpture, engraving and lithography; photographic works to which are assimilated works expressed by a process analogous to photography; works of applied art; illustrations, maps, plans, sketches and three-dimensional works relative to geography, topography, architecture or science [18].

Copyright laws vary from country to country, but as a rule do not provide less copyright protection than the Berne Convention, as long as the country is a member.

Software is not specially mentioned in the convention, but proprietary software is also protected by copyright law. The Software and Information Industry association (SIIA) and the Business Software Alliance (BSA) estimate that the value of pirated software worldwide has been approximately 11-13 billion US dollars per year for many years. Of course these figures are insecure as the activity is illegal, but the gap between the sales number for software licences and the calculated actual number of applications installed on computers gives some idea of the range of software piracy, as estimated by the SIIA. In China, for instance, dozens of factories are said to produce pirated CDs for export to other countries, even though laws exist to protect intellectual property rights, particularly for foreign work, but the laws are not enforced[17].

Fair Use

There is much discussion about the ethics of copying. A difficult area of copyright law is the so-called 'Fair Use'. The actual specifics of what is acceptable will be governed by national laws, and although similar, laws will vary from country to country. Cases dealing with fair use can be complex, as decisions are based on individual circumstances and judgments. To avoid problems, if in any doubt, it is advised to always get the permission of the owner, prior to use.

The UK Copyright Service provides copyright registration for original works by writers, musicians, artists, designers, software providers, authors, companies, organisations and individuals. Known as Copyright Witness internationally, and the UK Copyright Service in the UK, the service supports international copyright protection by securing independent evidence that will help prove originality and ownership in any future claims or disputes[19].

The UK copyright service defines fair use in the following way: *In copyright law, there is a concept of fair use, also known as; free use, fair dealing, or fair practice. Fair use sets out certain actions that may be carried out, but would not normally be regarded as an infringement of the work. The idea behind this is that if copyright laws are too restrictive, it may stifle free speech, news reporting, or result in disproportionate penalties for inconsequential or accidental inclusion.*

Under fair use rules, it may be possible to use quotations or excerpts, where the work has been made available to the public, (i.e. published). Provided that:

- The use is deemed acceptable under the terms of fair dealing.
- That the quoted material is justified, and no more than is necessary is included.

- That the source of the quoted material is mentioned, along with the name of the author.

Typical free uses of work include:

- Inclusion for the purpose of news reporting.
- Incidental inclusion.
- National laws typically allow limited private and educational use.

'Incidental inclusion' is where part of a work is unintentionally included. A typical example of this would be a case where holiday movie inadvertently captured part of a copyright work, such as some background music, or a poster that just happened to be on a wall in the background.

Copyright and the Internet

The Internet and 'public domain' are not synonymous. Work published on the Internet is not automatically placed in the public domain. In the US, material found on the web may be copied freely only if the information is created by the federal government, if the copyright has expired or the copyright has been abandoned by the holder.

The following example illustrates the problem: Google is planning to scan books and publish them through their service Google Print. This undoubtedly violates copyright protection. An author has two rights: The right to create a piece of work, and the right to publish that work. Scanning a book means creating copies of that book, and publishing the book on the Internet means making copies available for the public. The author must agree to this approach, there is no such thing as a silent agreement. Google's arguments are that this must be regarded as fair use, but in most European countries, there is no rule which makes the copying of books fair use. In all probability, it is not acceptable under US law either. Some countries have specific laws defining when it is legal to use parts of published work without the author's agreement, for instance for private and educational purposes. But making a piece of intellectual property public on the Internet is not covered by any of these regulations [20].

5.3 Patenting

A patent is related to inventions that are supposed innovative (or new), useful and *non-obvious*. An inventor may apply for a patent, and after the patent has been issued, the patent holder has an *exclusive right to commercially exploit the invention for a certain period of time (typically twenty years from the filing date of a patent application)*.

A patent gives an inventor the right over the invention for a limited period of time, restricting others from copying, selling or using the invention. Most patents today are for improvements in known technology.

Specific conditions must be fulfilled to get a patent. Major ones are that the invention must

- Be new or at least undocumented. The invention must not form part of the "state of the art", meaning everything that has been made available to the public before the date of applying for the patent. This includes published documents and articles, but also use, display, description.
- Involve an inventive step. As well as being new, the invention must not be obvious from the state of the art.
- Be industrially applicable. This condition requires that the invention can be made or used in any kind of industry.

[16].

5.4 Trade Marks

A trade mark is a *distinctive sign which is used to distinguish the products or services of one business from those of another business (Wikipedia)*.

A trade mark is *any sign which can distinguish the goods and services of one trader from those of another. A sign includes words, logos, colours, slogans, three-dimensional shapes and sometimes sounds and gestures. A trade mark is therefore a "badge" of trade origin. It is used as a marketing tool so that customers can recognise the product of a particular trader. To be registrable in the UK it must also be capable of being represented graphically, that is, in words and/or pictures*. [16]. Copying a logo or slogan is a violation of copyright law.

5.5 Designs

Design is defined as *product appearance, of the whole or a part of a product resulting from the features of, in particular, the lines, contours, colours, shape, texture or materials of the product itself or its ornamentation*. Design is usually protected by three legal rights: registered designs, unregistered design right, and artistic copyright.

Design registration gives the owner a monopoly on their product design, i.e. the right for a limited period to stop others from making, using or selling a product to which the design has been applied, or in which it has been incorporated, without their permission and is additional to any design right or copyright protection that may exist automatically in the design[16].

5.6 Privacy

The Internet with its millions of users, represents several threats to personal security and privacy. Sensitive information is published openly, including telephone conversations, electronic mail, trade secrets and health records. The technological development makes everybody vulnerable to unwanted snooping by governments, business competitors, terrorists, hackers and thieves, in addition to identity theft and fraud. *Privacy International (PI)* is a human rights group formed in 1990 as a watchdog on surveillance and privacy invasions by governments and corporations. PI is based in London, England, and has an office in Washington, D.C. PI has conducted campaigns and research throughout the world on issues ranging from wiretapping and national security, to ID cards,

video surveillance, data matching, police information systems, medical privacy, and freedom of information and expression[21].

The protection of privacy is strong in international law. Privacy is a fundamental human right, and one of the most important human rights of the modern age. It is protected in the Universal Declaration of Human Rights, the International Covenant on Civil and Political Rights, and in many other international and regional human rights treaties. Nearly every country in the world includes a right of privacy in its constitution. Modern constitutions include specific rights to access and control one's personal information. International agreements that recognize privacy rights are for instance the International Covenant on Civil and Political Rights or the European Convention on Human Rights, (full name: The Convention for the Protection of Human Rights and Fundamental Freedoms, Rome 4 November 1950).

Article 8 in this convention states that:

'Everyone has the right to respect for his private and family life, his home and his correspondence. There shall be no interference by a public authority with the exercise of this right except such as is in accordance with the law and is necessary in a democratic society in the interests of national security, public safety or the economic well-being of the country, for the prevention of disorder or crime, for the protection of health or morals, or for the protection of the rights and freedoms of others. '

Privacy International (PI) is a human rights group formed in 1990 as a watchdog on surveillance by governments and corporations. PI is based in London, England, and has an office in Washington, DC PI has conducted campaigns throughout the world on issues ranging from wiretapping and national security activities, to ID cards, video surveillance, data matching, police information systems, and medical privacy.

Privacy protection is also controlled by individual users. Internet users employ various programs and systems that provide privacy and security of communications. These include encryption, anonymous remailers, proxy servers and digital cash. Not all tools effectively protect privacy [21].

Phil Zimmermann and PGP - Pretty Good Privacy

Philip R. Zimmermann is the creator of Pretty Good Privacy, an email encryption software package. Zimmermann, a software engineer with more than 20 years of experience in cryptography and data security originally designed PGP as designed as a human rights tool, and published it for free on the Internet in 1991. As a consequence, Zimmermann was subjected to three years of criminal investigation, because the US government maintained that US export restrictions for cryptographic software were violated when PGP spread worldwide. However, the government dropped the case in 1991, and PGP has since become the most popular email encryption software in the world. Zimmermann founded PGP Inc. in 1991, a company that was acquired by Network Associates Inc (NAI) in December 1997. In 2002 PGP was acquired from NAI by a new company called PGP Corporation. Zimmermann is now a consultant for companies and industry on cryptography. His home page is found at <http://www.philzimmermann.com/>.

5.7 Privacy and Human Rights

The Universal Declaration of Human Rights from 1948 article 12 states that *'No one shall be subjected to arbitrary interference with his privacy, family, home or correspondence, nor to attacks upon his honour and reputation. Everyone has the right to the protection of the law against such interference or attacks.'*

According to the International Covenant on Civil and Political Rights - 1966 article 17:

1. *No one shall be subjected to arbitrary or unlawful interference with his privacy, family, home or correspondence, nor to unlawful attacks on his honour and reputation.*
2. *Everyone has the right to the protection of the law against such interference or attacks.*

Public information is defined as information knowable by all, for instance information you have provided to an organisation that has a right to share it with other organisations. An example of such information is the telephone directory. On the other hand, personal information is not part of a public record, for example your religious and political beliefs. If you personally disclose this information to an organisation with the right to inform other organisations, it becomes public information.

5.8 Oversight and Privacy

In countries with a data protection or privacy act, there is an official or agency that oversees enforcement of the act. The powers of these officials, Commissioner, Ombudsman or Registrar, vary by country. Several countries, including Germany and Canada, also have officials or offices on a state or provincial level.

Under Article 28 of the EU Data Protection Directive, all European Union countries must have an independent enforcement body[22]. Under the Directive, these agencies are given considerable power: governments must consult the body when the government draws up legislation relating to the processing of personal information; the bodies also have the power to conduct investigations and have a right to access information relevant to their investigations; impose remedies such as ordering the destruction of information or ban processing, and start legal proceedings, hear complaints and issue reports. Several countries that do not have a comprehensive act still have a commissioner. A major power of these officials is to focus public attention on problem areas, even when they do not have any authority to fix the problem. They can do this by promoting codes of practice and encouraging industry associations to adopt them. They also can use their annual reports to point out problems. For example, in Canada, the Federal Privacy Commissioner announced in his 2000 report the existence of an extensive database maintained by the federal government. Once the issue became public, the Ministry disbanded the database.

5.9 Freedom of Speech on the Internet: The Role of Freenet

'The Internet is a powerful and positive forum for free expression. Internet users, online publishers, library and academic groups

and free speech and journalistic organizations share a common interest in opposing the adoption of techniques and standards that could limit the vibrance and openness of the Internet as a communications medium. Indeed, content "filtering" techniques already have been implemented in ways inconsistent with free speech principles, impeding the ability of Internet users to publish and receive constitutionally protected expression.' [23].

Article 19 of the Universal Declaration of Human Rights and Article 10 of the European Convention on Human Rights are international human rights proclamations that support the freedom of speech, although implementation of this fundamental human right is unfortunately lacking in many countries. However, the right to freedom of speech is not unlimited, many governments may prohibit certain forms of expression. According to international law, restrictions on free speech must be provided by law, pursue an aim recognized as legitimate, and be necessary for the accomplishment of that aim. Some of the aims that are considered legitimate are protection of the rights and reputations of others and the protection of national security and public order, health and morals. There are differences of opinion among people of different nations and cultures as to when restriction of free speech meets these criteria [24].

The Internet has opened new possibilities for achieving freedom of speech, giving anybody the opportunity to publish whatever they want independent of legal measures. Pseudonymity and data havens (such as Freenet[25]) allow unlimited free speech, because the technology guarantees that material cannot be removed or censored.

'Freenet is free software which lets you publish and obtain information on the Internet without fear of censorship. To achieve this freedom, the network is entirely decentralized and publishers and consumers of information are anonymous. Without anonymity there can never be true freedom of speech, and without decentralization the network will be vulnerable to attack.'

Communications by Freenet nodes are encrypted and are routed through other nodes to make it extremely difficult to determine who is requesting the information and what its content is. Users contribute to the network by giving bandwidth and a portion of their hard drive (called the "data store") for storing files. Unlike other peer-to-peer file sharing networks, Freenet does not let the user control what is stored in the data store. Instead, files are kept or deleted depending on how popular they are, with the least popular being discarded to make way for newer or more popular content. Files in the data store are encrypted to reduce the likelihood of prosecution by persons wishing to censor content.

Websites which are censored by national governments are often re-hosted on a server in a country with no censorship restrictions. The United States has in many ways the world's least restrictive governmental policies on freedom of speech. Many websites re-host their content on an American server to escape censorship. This is especially the case with neo-nazi and other sites promoting racial hatred, since these are illegal in many European countries.

Another organization dedicated to protecting freedom of speech on the Internet is The Electronic Frontier Foundation (EFF), found at www.eff.org. The

'EFF is a donor-funded nonprofit and depends on your support to continue successfully defending your digital rights. Litigation is particularly expensive; because two-thirds of our budget comes from individual donors, every contribution is critical to helping EFF fight, and win, more cases.'

5.10 Google and China

The Chinese government has imposed Internet censorship in order to control or eliminate access to information on controversial and sensitive topics such as the Tiananmen Square protests of 1989, Falun Gong, Tibet, Taiwan, pornography or democracy. They have also enlisted the help of some American companies like MSN, who have subsequently been criticized by proponents of freedom of speech.

Online search giant Google launched a China-based search engine that will be self-censored to avoid posting results that antagonize China's communist government.

Google.cn uses the Chinese Web suffix *.cn* in addition to the existing Chinese-language Google website hosted on servers in the USA.

'In order to operate from China, we have removed some content from the search results available on Google.cn, in response to local law, regulation or policy. Removing search results is inconsistent with Google's mission, but providing no information is more inconsistent with our mission.'

Senior policy counsel Andrew McLaughlin, Google.

The Google mirror *elGoog* makes it possible to search for words like 'democracy' from inside China. <http://elgoog.rb-hosting.de/index.cgi>.

5.11 Transborder Data Flows

Data protection laws can be circumvented by transferring personal information to third countries, where the national law of certain countries do not apply. The data can then be processed in these countries, called 'data havens'. This is why most data protection laws include restrictions on the transfer of information to third countries unless the information is protected in the destination country. For example, Article 12 of the Council of Europe's 1981 Convention places restrictions on the transborder flows of personal data [22]. Similarly, Article 25 of the European Directive imposes an obligation on member States to ensure that any personal information relating to European citizens is protected by law when it is exported to countries outside Europe. It states:

'The Member States shall provide that the transfer to a third country of personal data which are undergoing processing or are intended for processing after transfer may take place only if the third country in question ensures an adequate level of protection.'

Therefore, there is currently growing pressure outside Europe for the passage of strong data protection laws. Those countries that refuse to adopt privacy laws may be unable to conduct certain types of information flows with Europe, particularly if they involve sensitive data. The European Commission determines

a third country's system for protecting privacy, where the main principle in this determination process is that the level of protection in the receiving country must be *adequate* rather than *equivalent*. In this way, privacy protection is guaranteed from the third party, although the precise dictates of the Directive are not strictly followed.

6 Educational aspects

6.1 Teaching ethics

In order for computer professionals to incorporate an *ethical responsibility*, it has been stressed that it is important to have this topic introduced to them already during their educational years.

As system administration is not a common educational programme yet, most of the topics on this issue have been inspired by the more general field of Computer Science and Information Systems. However, most of these issues are also relevant for the system administration education.

There has been significant controversy in the field of teaching ethics to computer science graduates. There have been debates on all imaginable aspects of the issue, on what kind of *subjects* should be covered by the curriculum, *how* the material should be integrated (separate course or modules in other courses), and *who* should actually teach these subjects.

However, *how* to incorporate social and ethical aspects into the technical programmes has not proven trivial. Common problems have been

- *What* should the students learn?
- *Who* should teach these untraditional subjects?
- *How* should these issues be integrated in the traditional curriculum?

Course Content and Learning Objectives

The *ACM/IEEE Computing Curriculum 1991*, demanded an inclusion of ethical and social aspects in the Computer Science curriculum, therefore, the project *ImpactCS* was funded [26]. The objective of this project was to define the core content and develop a framework for integrating social and ethical topics into the computer science curriculum [27].

The project delivered three reports [28], where each of them covered different aspects of the process of including ethics in the CS curriculum. Among the final results was a very broad list of topics, which included the following *learning objectives*:

- *Responsibility of the Computer Professional for Computer Science*. This topic again contains the subtopics
 - History of the development and impact of computer technology
 - Why be ethical?
 - Major ethical models
 - Definition of computing as a profession

- *Basic Elements of Ethical Analysis for Computer Science.* The elements covered by this section are
 - Ethical claims can and should be discussed rationally.
 - Ethical choices cannot be avoided.
 - Easy ethical approaches are questionable.
- *Basic Skills of Ethical Analysis for Computer Science.*
 - Arguing from example, analogy, and counter-example
 - Identifying stakeholders in concrete situations
 - Identifying ethical issues in concrete situations
 - Applying ethical codes to concrete situations
 - Identifying and evaluating alternative courses of action
- *Basic elements of Social Analysis for Computer Science.*
 - Social context influences the development and use of technology
 - Power relations are central in all social interactions
 - Technology embodies the values of the developers
 - Populations are always diverse
 - Empirical data are crucial to design and development processes
- *Basic Skills of Social Analysis for Computer Science.*
 - Identifying and interpreting the social context of a particular system
 - Identifying assumptions and values embedded in a particular system
 - Using empirical data to evaluate a particular implementation of a technology

As the list shows, the project ImpactCS provides a very broad range of subjects, not supported by all members of this community. David Preston claims this list is too long, and that most of the subjects are out of scope for an ethics course to be interesting for the Computer Science students [29]. A more narrow approach is recommended, to assure better relevance and hence interest from the students. In particular, they recommend spending considerable time focusing on the students' *own* experiences, as these tend to arouse more interest and participation among the students.

6.2 How Should Professional Ethics be Taught?

After establishing the learning objectives, there is still the issue of how to determine the teaching form for this kind of material. Using *case studies* is a very common approach, as this is claimed to improve the understanding of certain ethical dilemmas and increase interest and hence participation among the students. Some authors object to this, with the argument that case studies do not bring sufficient realism to the issues, and that the students should be encouraged to bring their *own* experiences to class [29].

The form of the lectures is also heavily debated, although it seems to be agreed upon that a mixture of *group discussions*, *guest lectures*, student-prepared *panel debates* and so on, is to be preferred over more traditional teacher-led lectures. One particular issue is whether the communication between teacher and students should be *one-way*, or if it should be more negotiation-based. Regarding so-called *open sessions*, where external speakers give talks with interaction with the students, there seem to be very difficult to predict which topics appeal to the students.

There is also significant emphasis on using a variety of technical resources in the classes, for instance local, national or international newspapers, local leaflets, film, music, student-generated material, fiction etc.

Who should teach ethics?

One of the problems with integrating an ethics and social aspects course in the computer science or system administration curriculum, is the lack of knowledge of the teachers and faculty. The discussion has been about who should teach these subjects: faculty of engineering or teachers of philosophy or social science.

The clear advantage of having teachers from the computer science community teach computer ethics is that they are familiar with the technological aspects of the use of computers, and the work situations in which ethical dilemmas arise. The use of case studies from industry is a recommended approach, and computer professionals have first-hand knowledge of the issues that are described in well-known cases and examples. Teachers of Philosophy may tend to present ethical theory and cases in a too theoretical and academic manner, which may have little relevance for computer professionals.

6.3 Teaching Students to Hack: A Necessity or Security Risk?

The emergence of diverse security threats has lead to increased focus on information security courses in the computer science education. A significant part of such courses has been the *security labs*, where students can achieve hands-on experience with important tools and concepts within information security, both regarding potential threats and how to protect the systems from diverse attacks [30].

Typical student activities in these labs are

1. Hacking (accessing systems or networks where the students do not have access),
2. Find vulnerable systems and penetrate them,
3. Remove evidence of penetration.

The argument for teaching the students such skills have been that the system administrators must possess the same skills as the attackers in order to be able to protect the system from the attacks (know your enemy). The reason for this is mainly the need for so-called *ethical hackers*.

The term *ethical hacking* comes from the practice of skilled 'hackers' trying to break into computer systems with the goal of discovering potential vulnerabilities in the system, in order to prevent attacks. More precisely, the ethical hackers aim at answering the following questions [31]:

- What part of the system is visible to an intruder?
- What can an intruder do with that information?
- Does anyone at the target notice the intruder's activities?

Arguments in favour of this practice has been [30]

- "that hacking skills are equivalent to audit skills as both are designed to discover flaws in the protection of data and and secure operation of a system. Just as auditors test systems for security or operational flaws, hackers "test" systems through attack;
- Knowledge of hacking skills and practice in attacking secured systems improves security by informing network administrators of how an exploit can be executed; and
- To provide the best security defense, a systems administrator must possess the same skills as the attacker."

This kind of work obviously requires people with significant skills and experience, which has lead to several universities and colleges including 'hacking' exercises as part of their computer security curriculum. Examples of such exercises have been writing port scanners, propagating viruses and exploit programs, in addition to activities like packet sniffing and injecting packets.

However, recently there has been increased focus on the downsides of this kind of organized student activity. There obviously is a significant risk of actually *educating* hackers or the "bad guys". Further, hacking skills does not necessarily make system administrators able to prevent such attacks.

To avoid malicious behaviour from students, some universities offer security labs only at the graduate level, with the motivation that only the most mature, and hence more responsible students will have access to this kind of information. Other universities have tried avoiding hands-on content or keeping the labs away from student access.

These methods have proved to have weaknesses, so as alternative methods for educating the students, the following has been suggested [30]:

- Teaching recovery activities
- Intensive vulnerability assessments of a simulated corporate network
- Network forensic investigations

The following questions are asked by the authors of [30]:

- How should students be taught to implement network and desktop security?
- Is hacking and virus-writing a pre-requisite for developing strong technical skills in detecting malicious activity?

- What course content (if any) should be off-limits?
- Will hacking skills for network administrators necessarily improve security and their employability?

7 Summary

The use of computers and computer networks poses ethical questions that often differ from our every-day ethical problems, mainly because of the nature of the technology itself. Computers have become universal tools, computers and networks are everywhere, shaping our lives, work and leisure time. With the rapid development of computer technology, new situations constantly arise and create policy vacuums, because the answers to ethical dilemmas are not clear. The law will not always be a help, because laws are designed to fit the technologies and problems of the day, and because technology is developing at such a tremendous speed. It takes time for the legal issues to catch up with the technical possibilities.

System administrators are often faced with ethical dilemmas in situations where they must make decisions, and take proper action. There are many paths to becoming a system administrator, and many lack formal education. Decisions may therefore often be founded on intuition, or be taken under pressure. The consequences can be serious, like the risk of losing a job, either through negligence, or because of being pressured into some illegal or unethical activity. For a system administrator it is therefore of the greatest importance to be acquainted with the rules, laws and policies regarding the performance of the work duties.

A system administrator must be able to make sound decisions based on reflection, knowledge, common sense and professional responsibility, and is, according to the sage code of Ethics, also obliged to educate himself/herself on technical, legal and ethical issues. In this chapter we have sought to outline some of the most important laws, ethical principles and guidelines that may assist the system administrator in fulfilling his or her professional duties.

References

- [1] Paul N. Edwards Christopher A. Lee, Brian S. Williams. Evaluating information and communications technology: Perspectives for a balanced approach, 2001.
- [2] Michael J. Quinn. *Ethics for the Information Age*. Pearson Education, 2006.
- [3] James P. Sterba. *Ethics: The Big Questions*. Blackwell Publishers, 2002.
- [4] Free Software Foundation (FSF). <http://www.fsf.org/>.
- [5] Wikipedia, 13.08.2006. <http://www.wikipedia.org>.
- [6] Suggested Guidelines for Writing a Code of Ethics/Conduct. <http://www.deloitte.com/dtt/cda/doc/content/Suggested%20Guidelines%20for%20writing%20a%20code%20of%20ethics%2C%20conduct.pdf>.

- [7] The ACM Software Engineering Code of Ethics and Professional Practice. <http://www.acm.org/constitution/code.html>.
- [8] IEEE. Code of ethics. <http://www.ieee.org/portal/pages/about/whatis/code.html>.
- [9] SAGE. Code of ethics. <http://www.sage.org/ethics>.
- [10] Online Ethics Centre. <http://onlineethics.org/cite-link.html>, September 2006.
- [11] The Online Ethics Center for Engineering and Science. <http://www.onlineethics.org>.
- [12] Online Ethics Helpline. <http://onlineethics.org/helpline/index.html>, September 2006.
- [13] Scenarios about computers and software. <http://onlineethics.org/com/scenarios.html>.
- [14] Protection for Whistleblowers. <http://www.ignet.gov/randp/f01c10.pdf>.
- [15] Online Ethics Centre. <http://onlineethics.org/moral/austin/index.html>, September 2006.
- [16] Intellectual Property. <http://www.intellectual-property.gov.uk/index.html>.
- [17] Sara Baase. *A Gift of Fire*. Pearson Education, 2003.
- [18] Berne Convention for the Protection of Literary and Artistic Works. http://www.wipo.int/treaties/en/ip/berne/trtdocs_wo001.html.
- [19] The UK Copyright Service. <http://www.copyrightservice.co.uk/>.
- [20] Olav Torvund. Copyright: An introduction - (Norwegian). <http://www.torvund.net/artikler/art-opphav.asp>.
- [21] Privacy International. <http://www.privacyinternational.org>.
- [22] EU Data Protection Directive. <http://www.opsi.gov.uk/ACTS/acts1998/19980023.htm>.
- [23] Internet Free Expression Alliance. <http://www.ifea.net>.
- [24] Wikipedia. http://en.wikipedia.org/wiki/Freedom_of_speech.
- [25] Freenet project. <http://freenetproject.org/>, September 2006.
- [26] The ImpactCS Home Page. <http://www.seas.gwu.edu/impactcs/>.
- [27] Laurie H. Werth. Getting started with computer ethics. In *Proceedings of the twenty-eighth SIGCSE Technical Symposium on Computer Science Education*, pages 1–5. ACM Press, 1997.
- [28] C. Dianne Martin and Elaine Yale Weltz. From awareness to action: Integrating ethics and social responsibility into the computer science curriculum. *Computers and Society*, 1999.

- [29] David Preston. What makes professionals so difficult: An investigation into professional ethics teaching. In *Proceedings of POLICY'98: Ethics and Social Impact*. ACM Press, 1998.
- [30] Patricia Y. Logan and Allen Clarkson. Teaching students to hack: Curriculum issues in information security. *SIGCSE, ACM*, 2005.
- [31] C.C. Palmer. Ethical hacking. <http://www.research.ibm.com/journal/sj/403/palmer.html>, 2001.